IBM

eserver

iSeries

CICS for iSeries Problem Determination

Version 5

SC41-5453-00

IBM

@server

iSeries

CICS for iSeries Problem Determination

Version 5

SC41-5453-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix C, "Notices" on page 103.

First edition (September 2002)

This edition applies to version 5 release 2 modification 0 of the IBM licensed program CICS for iSeries, program number 5722-DFH, and to all subsequent releases and modifications until otherwise indicated in new editions. This edition applies only to reduced instruction set computer (RISC) systems.

This edition replaces SC33-1384-00.

Contents

About CICS for iSeries Problem Determination (SC41-5453)

This book contains information relevant to problem determination and resolution.

Who should read this book

This book is intended for the person responsible for CICS problem determination.

The reader is assumed to have general data processing knowledge with an understanding of CICS, and an understanding of or access to information about the iSeries server. The CICS™ library does not attempt to teach the fundamentals of CICS. Extensive systems programming experience should not be necessary.

This book is organized into 3 parts:

- Part 1, "Approach to Problem Determination" on page 1
- Part 2, "Dealing with the Problem" on page 23
- Part 3, "Using traces and dumps in problem determination" on page 67

Conventions and terminology used in this book

The term 'CICS' refers to 'CICS/400' unless otherwise stated.

MB equals 1 048 576 bytes.

KB equals 1024 bytes.

Prerequisite and related information

Use the iSeries Information Center as your starting point for looking up iSeries technical information.

You can access the Information Center two ways:

- From the following Web site:

  `http://www.ibm.com/eserver/iseries/infocenter`
- From CD-ROMs that ship with your Operating System/400 order:

  iSeries Information Center, SK3T-4091-02. This package also includes the PDF versions of iSeries manuals, *iSeries Information Center: Supplemental Manuals*, SK3T-4092-01, which replaces the Softcopy Library CD-ROM.

The iSeries Information Center contains advisors and important topics such as Java, TCP/IP, Web serving, secured networks, logical partitions, clustering, CL commands, and system application programming interfaces (APIs). It also includes links to related IBM® Redbooks and Internet links to other IBM Web sites such as the Technical Studio and the IBM home page.

With every new hardware order, you receive the *iSeries Setup and Operations CD-ROM*, SK3T-4098-01. This CD-ROM contains IBM @server iSeries Access for Windows and the EZ-Setup wizard. iSeries Access offers a powerful set of client and server capabilities for connecting PCs to iSeries™ servers. The EZ-Setup wizard automates many of the iSeries setup tasks.

CICS for iSeries library

These books form the CICS for iSeries library that is delivered with the product:

CICS for iSeries Administration and Operations Guide, SC41-5455-00

This guide gives introductory information about CICS for iSeries. It then provides information about system and resource definition, setup of a system, and operator commands.

CICS for iSeries Application Programming Guide, SC41-5454-01

This manual provides programming guidance information, in narrative form with examples. This is followed by the reference section describing the syntax and use of each document.

CICS for iSeries Intercommunication, SC41-5456-00

This manual describes the CICS for iSeries side of communication between CICS® systems running on different platforms. There is a similar manual for each CICS platform.

CICS for iSeries Problem Determination, SC41-5453-00

This manual provides guidance in problem determination for users of CICS for iSeries.

CICS Family: Interproduct Communication, SC34-6030-00

This manual, which is also part of the libraries of the other CICS family members, gives an overview of communications between CICS systems running on different platforms.

CICS Family: API Structure, SC33-1007-02

This manual, which is also part of the libraries of the other CICS family members, gives a quick reference to the level of support that each member of the CICS family gives to the CICS application programming interface. It is designed for customers and software vendors developing applications able to run on more than one CICS platform and porting applications from one platform to another.

Books from related libraries

COBOL

COBOL/400 User's Guide, SC09-1812-00

COBOL/400 Reference, SC09-1813-00

C

ILE Concepts, SC41-5606-06

Control language (CL)

CL Programming, SC41-5721-05

Control Language topic in the iSeries Information Center

Problem determination

Performance Tools for iSeries, SC41-5340-01

Work Management, SC41-5306-03

Basic Operations topic in the iSeries Information Center

Miscellaneous

File Management topic in the iSeries Information Center.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other iSeries documentation, fill out the readers' comment form at the back of this book.

- If you prefer to send comments by mail, use the readers' comment form with the address that is printed on the back. If you are mailing a readers' comment form from a country other than the United States, you can give the form to the local IBM branch office or IBM representative for postage-paid mailing.
- If you prefer to send comments by FAX, use either of the following numbers:
  - United States, Canada, and Puerto Rico: 1-800-937-3430
  - Other countries: 1-507-253-5192
- If you prefer to send comments electronically, use one of these e-mail addresses:
  - Comments on books:

    RCHCLERK@us.ibm.com
  - Comments on the iSeries Information Center:

    RCHINFOC@us.ibm.com

Be sure to include the following:

- The name of the book or iSeries Information Center topic.
- The publication number of a book.
- The page number or topic of a book to which your comment applies.

Part 1. Approach to Problem Determination

Chapter 1. Introduction to problem determination

This chapter helps you to decide where to begin looking for causes when you have a problem with your CICS/400 system. Usually, you start with a symptom, or set of symptoms, and trace them back to their cause. This process is called **problem determination**.

This chapter covers:

- "Problem determination and problem solving"
- "How this book can help you"
- "Preliminary checks" on page 4
- "What to do next" on page 8.

Problem determination and problem solving

Problem determination is not to be confused with problem solving, although it is true that often in the process of problem determination, you obtain enough information to enable you to solve your problem. Examples of situations where this can happen are:

- End-user errors
- Application programming errors
- System programming errors, such as in resource definitions.

However, you may not always be able to solve a problem yourself after determining its cause. For example:

- A performance problem may be caused by a limitation of your hardware.
- You may find that the cause of a problem is in the CICS/400 code. If this happens, you will need to contact your IBM Support Center for a solution.

How this book can help you

This book starts with the symptoms of the problem, and then uses these symptoms to classify the problem. For each class of problem, possible causes and techniques are suggested to assist you.

Always assume first that the problem has a simple cause (for example, an application programming error). If, as a result of investigation, you find that the cause of the problem is not straightforward, go on to consider causes that may be more complex and therefore more difficult to determine. Having exhausted all other possibilities, consider the possibility that the cause of the problem may be in the CICS/400 code itself.

Table 1 on page 4 shows you what to read first.

Table 1. Where to look first

Do you want guidance about resolving a specific CICS problem?	No→	Do you want guidance about useful sources of information?	No→	Do you want guidance about using traces and dumps?	No→	Do you want guidance about IBM Support Centers or APARs?
↓Yes		↓Yes		↓Yes		↓Yes
		See Chapter 3		See Chapter 11, Chapter 12, and Chapter 13		See the Basic Operations topic under Systems Management in the Information Center
Have you done any preliminary checks?	No→	See 4				
↓Yes						
Have you classified the problem?	No→	See Chapter 2				
↓Yes						
Is it an ABEND?	Yes→	Is it a transaction ABEND?	Yes→	See Chapter 4		
		↓No				
↓No		See Chapter 5				
Is it a WAIT?	Yes→	See Chapter 6				
↓No						
Is it a LOOP?	Yes→	See Chapter 7				
↓No						
Is it a PERFORMANCE problem?	Yes→	See Chapter 8				
↓No						
Is it an INCORRECT OUTPUT problem?	Yes→	See Chapter 9				
↓No						
Is it a STORAGE VIOLATION problem?	Yes→	See Chapter 10				
↓No						
Refer to Chapter 2						

Preliminary checks

Before starting problem determination in detail, it is worth checking to see whether there is an obvious cause of the problem, or a likely area in which to start your investigation. This approach to debugging can often save time by highlighting a simple error, or by narrowing down the range of possibilities.

The sections that follow list some fundamental questions that you need to consider. As you go through the questions, make a note of anything that might be relevant to the problem. Even if the observations you record do not at first suggest a cause, they could be useful to you later if you need to carry out systematic problem determination.

Has the CICS system run successfully before?

If the CICS system has not run successfully before, it is possible that you have not yet set it up correctly. You need to refer to the *CICS for iSeries Administration and Operations Guide* for guidance on doing this.

Are there any messages explaining the failure?

If a transaction abends, and the task terminates abnormally, CICS sends a message reporting the fact to a number of areas on the iSeries. Within CICS/400, the abend message is sent to the CSMT log, if a CSMT queue has been defined in the CICS control region where the transaction was executed. In addition, the message is written to these message destinations:

- Job log for the job executing the transaction (the process that initially invoked the STRCICSUSR CL command)
- Job log of the CICS/400 control region (the process that initially invoked the STRCICS CL command)
- QSYSOPR (or *SYSOPR) message queue
- QHST log
- The CICS/400 terminal

If you find a message or a series of messages in these areas, these might immediately suggest a reason for the failure.

Were there any unusual messages associated with the startup of the CICS control region or while the control region was running before the error occurred? These might indicate some system problem that prevented your transaction from running successfully.

If you see any messages that you don't understand, you can display a help panel. Place the cursor on the message and press F1. Alternatively, you can use the DSPMSGD CL command (display message description). Make sure you view the second-level text for an explanation and, perhaps, a suggested course of action that you can take to resolve the problem.

Note: When error messages are examined, keep a record of each message plus return codes, sense codes, log identifiers, error classes, and any other relevant items that are detailed in the messages. To and From programs, with associated instruction numbers, may be logged with the messages. Module names, with associated instruction (or statement) numbers, may be recorded in the various logs. All of these entries form basic and essential information if the IBM Support Center becomes involved. If there is any possibility that the information might be erased from the system, it is worth considering saving the information to tape.

Can the failure be reproduced?

If the failure is reproducible, consider the conditions under which it can be reproduced:

- Can you identify any application that always seems to be in the system when the problem occurs? If so, examine the application to see whether it is in error. Apart from application coding errors, consider whether you have defined sufficient resources. Typically, if you had not defined sufficient resources, you would find that the problem is related to the number of users of the application. For example, the physical file space may have run out and cannot be extended.

- If the problem is not related to any particular application, consider your CICS system definition parameters. Experience has shown that poorly defined parameters are often the cause of problems in CICS systems. You can find guidance about setting up your CICS system in the *CICS for iSeries Administration and Operations Guide*.
- Does the problem seem to be related to system loading? A sign of this might be that the symptoms occur at specific times of day, when activity on the system peaks. If so, the system might be running near its maximum capacity, or it might be in need of tuning. For guidance on dealing with this sort of problem, see *Performance Tools for iSeries*.

Is the failure intermittent?

If you have an intermittent failure, particularly one that does not always show the same symptoms, the problem is likely to be very difficult to resolve. An intermittent failure can be caused by an interrupted update. Furthermore, the transaction that caused the error might have been deleted from the system long before the symptoms are seen.

A method you can use to investigate random overlays is described in Chapter 10, "Dealing with storage violations" on page 61.

Have any changes been made since the last successful run?

When you are considering changes that might recently have been made, think about the CICS system, and also about the other programs it interfaces with, the hardware, and any new applications. Consider also the possibility that a new application that you don't yet know about might have been run on the system.

- If a program temporary fix (PTF) has been applied to CICS, check whether an error message was issued on installation. If the installation was successful, check with the IBM Support Center for PTF errors that may have been raised.

  If a PTF has been applied to any other program, consider the effect it might have on the way CICS interfaces with it.
- If a **hardware modification** has been carried out, it is possible that your CICS system does not function as it did before the change.
- If a **new or modified application** has just been included, check for error messages in the output from the following sources:
  - Translator
  - Compiler
- If your **initialization procedure** has been changed, for example by CL commands, CICS system initialization parameters, or an iSeries device definition, consider whether that might be the cause of the problem. In addition, check for error messages sent to the console during initialization.

Are specific parts of the network affected by the problem?

You might be able to identify specific parts of the network that are affected by the problem. If you can, look for any explanatory message from the access method. Even if no message has been sent to the console, you might find one in the job log.

Have you made any network-related changes that might account for the problem?

If a single terminal is affected, there could be an error in the terminal definition. Consider both the CICSDEV definition, and the DEVTYPE definition it uses. See the *CICS for iSeries Administration and Operations Guide*.

If a number of terminals are affected, try to identify some factor that is common to all of them. For example:

- Do they use the same DEVTYPE definition? If so, it is likely that there is an error in that DEVTYPE definition.
- Is the whole network affected? If so, CICS has probably stalled. See "What to do if CICS has stalled" on page 36 for advice about dealing with CICS system stalls.

For more information about configuring CICS/400 for the network, see *CICS for iSeries Intercommunication*.

You may be able to identify specific parts of the network affected by the problem by examining the messages in the CSMT log, or in the other message destinations referred to previously. In addition, the QSYSOPR message queue may contain Operating System/400® (OS/400) messages issued by the operating system on behalf of CICS. If you know the device name, or the naming conventions for devices on your system, you can use the WRKCFGSTS CL command to display devices and then look at the status of these devices.

Has the application run successfully before?

If the problem appears to involve one particular application, consider whether the application has run successfully before.

Before you answer **YES** to this question, consider these subsidiary questions:

- **Have any changes been made to the application since it last ran successfully?**

  If so, it is likely that the error lies somewhere in the new or modified part of the application. Examine the changes to see whether you can find an obvious reason for the failure.

  Ensure that resource definition changes required for the application changes have been installed in the system running the application.

  If any maps have been changed, be sure that you created (by using the CRTCICSMAP command) a new *USRSPC object for the map, and that all application programs using the new map have been compiled using the new logical map (COBOL copybook or ILE C header file) created as a result of the command.

- **Have all the functions of the application been fully exercised before?**

  Could the failure have occurred when part of the application that had never been invoked before was used for the first time? If so, the error probably lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

  If a program has been run successfully on previous occasions, examine the contents of any records, screen data, PF keys, or files that were being processed when the error occurred. They may contain some unusual data values, causing a rarely used path in the program to be invoked.

  Check whether the records required at the time of the error can be retrieved successfully. If there is more than one method of accessing the information, make sure you test whether the data can be accessed in precisely the way the program attempts to access it.

  Check all fields within the records at the time of the error to see whether they contain data in a format acceptable to the program.

Has the application not run successfully before?

If your application has not yet run successfully, you need to examine it carefully to see whether you can find any errors.

Before you look at the code, examine the output from the **translator** and **compiler** (the CRTCICSCBL CL command for COBOL/400® programs or the CRTCICSC CL command for C programs) to see whether any errors have been reported. If your program failed to translate or compile, it will also fail to run if you attempt to invoke it.

If the evidence shows that each of these steps was accomplished without error, you must consider the coding logic of the application. Do the symptoms of the failure indicate the function that is failing, and therefore the piece of code in error? The following is a list of some programming errors commonly found in applications:

- CICS areas are addressed incorrectly
- Transient data is managed incorrectly
- Temporary storage is managed incorrectly
- File resources are not released
- Storage is corrupted by the program
- Return codes from CICS requests are ignored

In addition, if you are porting the application from a mainframe CICS system, you should check that the program variables are defined with the correct lengths. If the lengths are not of the correct size, storage overwrites can occur. On mainframe CICS systems, information associated with an array is held immediately before the array, which means that a storage overwrite may remain undetected. With CICS/400, information associated with the array is held after the array, which means that a storage overwrite is almost certain to have an effect.

What to do next

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you should now be able to resolve it, possibly with the help of other parts of this book, other books in the CICS/400 library, or the personnel in your IBM Support Center.

If you have not yet found the cause, you must start to look at the problem in greater detail. Begin by finding the best category for the problem, using the approach described in Chapter 2, “Classifying the problem” on page 9.

Chapter 2. Classifying the problem

The purpose of this chapter is to help you classify your problem into one of the categories used by the IBM support center for its service procedures. IBM support center staff have found that classifying the problem first is a good approach to problem determination.

This chapter covers:

Symptom keywords for classifying problems

IBM keeps records of all known problems with its licensed programs on RETAIN®, which is a worldwide, online, fully searchable database. IBM support center staff continually update the database as new problems are reported, and they regularly search the database to see whether problems they are told about are already known.

If you have the IBM INFORMATION/ACCESS licensed program, 5665-266, you can look on the RETAIN database yourself. Each of the problems there has a classification type.

Electronic Customer Support is used on the iSeries to obtain information about problems that are already known.

For software problems, the classifications are:

- ABEND
- WAIT
- LOOP
- POOR PERFORMANCE (or PERFM)
- INCORRECT OUTPUT (or INCORROUT)
- MESSAGE.

All but the last of these, MESSAGE, are considered in this section. If you receive a CICS error message, you can get help by putting the cursor on the message and pressing F1. If you cannot access a CICS/400 screen, you can use the DSPMSGD CL command (display message description) with the message file *library name*/AEGMSG. Be sure to view the second-level text. If you get a message from another IBM program, or from the operating system, you need to look at the message for an explanation of what that message means.

The message might give you enough information to solve the problem, or you may need to return to this chapter for further guidance. If you are unable to understand the message, you may need to contact your IBM Support Center for help.

In addition to the RETAIN classifications used by the IBM Support Center, this section considers that a STORAGE VIOLATION belongs to its own problem class.

Confirming that you have a storage violation can be difficult, and unless you get a CICS message stating explicitly that this is the problem, you could get almost any symptom, depending on what has been overlaid. You might, therefore, classify it initially as one of the RETAIN symptom types described earlier in this section.

One sort of problem that might give rise to a number of symptoms, usually ill-defined, is that of poor application design. See the *CICS for iSeries Application Programming Guide* for guidance in this area.

Using the symptoms to classify the problem

The headings that follow are to help you to classify the problem on the basis of the symptoms you observe.

The symptoms might enable you to classify the problem correctly at once, but sometimes you may need to consider the evidence carefully before coming to a decision. You might need to make a "best guess", and then be prepared to reconsider later on the basis of further evidence.

Look for the section heading that most nearly describes the symptoms you have, and then follow the advice given there.

CICS has stopped running

There are three main reasons why CICS might unexpectedly stop running:

1. There could be a CICS system abend. For example, the CICS control region has abended.
2. The CICS control region could be in a wait state.
3. A job in the system is monopolizing resources.

Consider, too, the possibility that CICS might still be running, but only slowly. Be certain that there is no activity at all before carrying out the checks in this section. If CICS *is* running slowly, you probably have a performance problem. If so, read "CICS is running slowly" on page 11 to confirm this before going on to "Where to look next" on page 17 for further advice.

If CICS really has stopped running, look for any message that might explain the situation. Begin by looking at the following destinations for messages:

- **The OS/400 QSYSOPR message queue**. Look for any message saying that the CICS job has abnormally terminated. If you find one, it means that a CICS system abend has occurred and that CICS is no longer running. Few messages are written to this queue, but finding a message here indicates a serious problem.

  If you don't find any explanatory message in the QSYSOPR message queue, check in the job logs for control region or user shell messages.
- **The job log**. This is a message queue attached to a particular OS/400 job. Each control region and shell is considered a separate job on the OS/400. Job logs are created if one of the following ways:
  - Automatically if a CICS/400 or OS/400 job ends abnormally
  - The LOG parameter of the CHGJOB CL command is set to 4 00 *SECLVL

  If you find a message in the job log, use the help facility to make sure there has been a CICS *system* abend. Do this by placing the cursor on the message and pressing F1. If you cannot access a CICS/400 screen, use either the DSPMSGD CL command or the WRKMSG CL command.

If you see only a *transaction* abend message in the job log, that won't account for CICS itself not running, and you should not classify the problem as an abend. A faulty transaction could hold CICS up, perhaps indefinitely, but CICS would resume work again if the transaction abended.

- **QHST log file**. This file automatically receives a copy of any message sent to SYSOPR. This destination is a possible alternative to the job log, and may, therefore, receive messages that are not directed to the job log.
- **CSMT log**. This is an equivalent message queue to mainframe CICS. However, it is up to the system administrator to define this log as an intrapartition transient data queue. Therefore, it may not be available at all OS/400 sites. For information on setting up CSMT, see the *CICS for iSeries Application Programming Guide*.
- **User terminal and program message queue**. For an interactive job, a message may be written on the 24th line of the user's screen. Neither this destination nor the program message queue is used for problem determination.

Here is an example of a message that might accompany CICS system abends, and which you would find on the job log:

AEG1502 The CICS/400 control region *ctlrgn* **detected a serious error.**

If you get this message, or any others for which the system action is to terminate CICS, turn to Chapter 5, "Dealing with CICS system abends" on page 29 for advice on what to do next.

If you can find no message saying that CICS has terminated, it is likely that the system is in a wait state, or that a program is not allowing CICS to run. These two possibilities are dealt with in Chapter 6, "Dealing with waits" on page 31 and Chapter 7, "Dealing with loops" on page 39.

CICS is running slowly

If CICS is running slowly, it is likely that you have a performance problem. It could be because your system is badly tuned, or because it is operating near the limits of its capacity.

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly designed transaction could be the cause. You might classify the problem initially as "poor performance", but be prepared to reconsider your classification later.

The following are some individual symptoms that could contribute to your perception that CICS is running slowly:

- Tasks take a long time to start running.
- Some low priority tasks will not run at all.
- Tasks start running, but take a long time to complete.
- Some tasks start running, but do not complete.
- No output is obtained.
- Terminal activity is reduced, or has ceased.
- Intersystem communication (ISC) tasks won't run

Some of these symptoms do not, in isolation, necessarily mean that you have a performance problem. They could indicate that some task is in a loop, or is waiting on a resource that is not available. Only you can judge whether what you see

should be classified as “poor performance”, in the light of all the evidence you have. Refer to “Distinguishing between waits, loops, and poor performance” on page 14 for guidance.

You might be able to gather more detailed evidence by using the tools and techniques that the OS/400 provides for collecting performance data.

- CEMT. You can use this transaction to determine which tasks are running in the control region. However, you must remember that, because most transactions run pseudoconversationally, a transaction appears on the CEMT INQUIRE TASK screen only if it is in the process of communicating with CICS/400.
- Statistics. You can use these to gather information about the system as a whole, without regard to tasks.
- OS/400 monitoring. You can use this facility to collect information about tasks.
- OS/400 job tracing.
- CICS tracing.

These are not specific tools for collecting performance data, but you can use them to gather detailed information about performance problems.

For more information concerning CICS tracing, refer to the EXEC CICS commands related to tracing in the *CICS for iSeries Application Programming Guide*. In addition, refer to Chapter 12, “Using traces in problem determination” on page 73 and Chapter 13, “PRTCICSTRC (Print CICS Trace) command” on page 85.

If a task fails to start, look first in the job log of the job that was running the CICS task. An explanatory message there may describe the reason why the task was not started. If no messages are found there, and you are authorized, check the job log of the control region. In addition, the QSYSOPR message queue and the QHST message log may contain explanatory messages.

If no explanatory message is found, the task is possibly being prevented from starting due to a system overload problem. Otherwise, classify the problem tentatively as “poor performance”, and turn to Chapter 8, “Dealing with performance problems” on page 45 for further guidance.

A task is running slowly

If a task is running slowly, it is likely that the explanation lies with the task itself. It could be in a loop, or it could periodically be entering a wait state. You need to decide which of these possibilities is the most likely before starting systematic problem determination. Refer to “Distinguishing between waits, loops, and poor performance” on page 14 for guidance.

Note: Don’t overlook the possibility that the task might simply be doing unnecessary work that does not change the final result. For example, starting a skip sequential browse with large gaps between the keys, or failing to finish a browse because it is holding onto resources.

A task stops running at a terminal

When a task stops running at a terminal, you will notice either or both of these symptoms:

- No output is obtained at the terminal.
- The terminal accepts no input.

First, make sure that the task is still in the system. Use a CEMT INQ TASK command to check its status, and make sure that it has not simply ended without writing back to the terminal.

If the terminal has a display unit, check to see whether a special symbol has been displayed in the operator information area that could explain the fault. If the operator information area is clear, make sure no message has been sent to any of the possible message destinations used for error messages. The job log of the job that was running the CICS task is the most likely destination.

If you can find no explanation for the problem, the fault is probably associated with the task running at the terminal. These are the possibilities:

- The task is in a wait state.
- The task is in a loop.
- There is a performance problem.

Read "Distinguishing between waits, loops, and poor performance" on page 14 to find out which of these is the most likely explanation. You can then refer to the appropriate chapter for advice about dealing with the problem.

A transaction has abended

If the transaction abended when you ran your application, CICS gives you an error message on your screen as well as a message in the job log, CSMT, and QSYSOPR message queue. Use the message help facilities of the OS/400 to view the first- and second-level text of the message. In addition, the CICS/400 messages are contained in the QCICS/AEGMSG *MSGF. Messages are further discussed in Chapter 5, "Dealing with CICS system abends" on page 29. The WRKMSGF or DSPMSGD CL commands can be used to browse CICS/400 messages. For more information about transaction abends refer to Chapter 4, "Dealing with transaction abends" on page 25.

You have obtained incorrect output

Incorrect output might be regarded as any sort of output that you were not expecting. However, use the term with care in the context of problem determination, because it might be a secondary effect of some other type of error. For example, if you get any sort of repetitive output, it may be that looping is occurring even though that output is not what you had expected. Also, CICS sends messages in response to many errors that it detects. You might regard the messages as "incorrect output", but they are only symptoms of another type of problem.

If you have received an unexpected message, and its meaning is not clear, use the message help facility or the WRKMSGF or DSPMSGD CL commands for an explanation. The second-level message text might suggest a simple correction that you can make to the application or the system.

The types of incorrect output dealt with in this part are:

- Incorrect trace or dump data:
  - Wrong destination
  - Wrong type of data captured
  - Correct type of data captured, but the data values were unexpected
- Wrong data displayed on the terminal

You can find advice about investigating the cause of any of these in Chapter 9, “Dealing with incorrect output” on page 49.

A storage violation has occurred

In many cases, storage violations are difficult to detect in an application. You could, for example, get a program check because code or data has been overlaid. You might suspect some other type of problem at first, and only after proceeding with your investigation find that a storage violation has occurred.

When CICS detects that storage has been corrupted, this message is sent to the job log:

AEG0504 Storage control has detected a serious error.

If you see this message, or you know (through other means) that a storage violation has occurred, turn to Chapter 10, “Dealing with storage violations” on page 61 for advice about dealing with the problem.

Distinguishing between waits, loops, and poor performance

Waits, loops, and poor performance can be quite difficult to distinguish. In some cases you might need to carry out a detailed investigation before deciding which classification is the right one for your problem.

Any of the following symptoms could be caused by a wait, a loop, or by a badly tuned or overloaded system:

- One or more user tasks in your CICS system fails to start
- One or more tasks stays suspended
- One or more tasks fails to complete
- No output is obtained
- Terminal activity is reduced, or has ceased
- The performance of your system is poor.

You must attempt to classify the problem before adopting a solution strategy. To do this, carefully consider the available evidence.

This section gives you guidance about choosing the best classification. When you have decided on that, read the appropriate chapter as indicated later in this section for further advice. However, note that in some cases your initial classification could be wrong, and you then need to reappraise the problem.

Waits

For the purpose of problem determination, a wait state is regarded as a state in which the execution of a task has been suspended. That is, the task has started to run, but it has been suspended without completing and has subsequently failed to resume.

The task might typically be waiting for a resource that is unavailable. A wait might affect just a single task, or a group of tasks that may be related in some way. If none of the tasks in a CICS system are running, CICS is in a wait state. The way to handle that situation is dealt with in “What to do if CICS has stalled” on page 36.

If you are authorized to use the CEMT transaction, you can find out which user tasks or CICS-supplied transactions are currently suspended in a running CICS system by using the CEMT INQ TASK command. Use the transaction several times, perhaps repeating the sequence after a few minutes, to see whether any task stays suspended. Is it reasonable for that task to remain suspended for an extended period? Does the task use any resources that might suggest possible causes of the problem?

You can use an INQUIRE TASK command as an alternative to the CEMT transaction. You can execute this command under the command-level interpreter CECI, or from a transaction.

Use the EXEC CICS INQUIRE TASK command to find the task numbers of all user tasks. If you use this command repeatedly, you can see which tasks stay suspended. You may also be able to find some relationship between several suspended tasks, perhaps indicating the cause of the wait.

However, you should allow for the possibility that a task may stay suspended because of an underlying performance problem, or because some other task may be looping.

If you can find no evidence that a task is waiting for a specific resource, you should not regard this as a wait problem. Consider instead whether it is a loop or a performance problem.

If it seems fairly certain that your problem is correctly classified as a wait, and the cause is not yet apparent, turn to Chapter 6, “Dealing with waits” on page 31 for guidance on how to solve this problem.

Loops

A loop is the repeated execution of some code. If you have not planned the loop, or if you have designed it into your application but for some reason it fails to terminate, you get a set of symptoms that vary depending on what the code is doing. In some cases, a loop may at first be diagnosed as a wait or a performance problem, because the looping task competes for system resources with other tasks that are not involved in the loop.

The following are some characteristic symptoms of loops:

- **The ‘input inhibited’ symbol** is permanently displayed in the operator information area of a display unit, or stays displayed for long periods.
- **Processor usage is very high**. You can check processor usage for any OS/400 job by using the WRKACTJOB CL command to display the active users. If usage is much greater than it is for similar jobs, the job might be looping. You should also be aware that the WRKACTJOB command itself can use a lot of CPU, and you need to make allowances for this when assessing whether the job is looping.
- **There is reduced activity at terminals**, or possibly no activity at all.
- **One or more CICS control regions appear to be stalled**, or to be continuing only slowly.
- **No CICS messages are written** to any destination when they are expected.
- **No new shells can be started**.
- **Existing shells remain suspended**.
- **Repetitive output is obtained**. Try looking in these areas:
  - Terminals or the message destinations.

  - Temporary storage queues. You can use CEBR to browse them online. If you are not logged on to CICS/400, or if the current user shell is locked, you can use the DSPPFM CL command to browse the AAEGxxxxTR and AAEGxxxxTN files from outside the CICS/400 control region.
  - Data files and CICS journals
  - Trace user spaces.
- **The demand for storage is excessive**. If the loop contains a GETMAIN request, storage is acquired each time this point in the loop is passed, as long as sufficient storage to satisfy the request remains available. If storage is not also freed in the loop, CICS eventually goes short on storage.

  CICS issues a message prefixed by AEG06xx when storage space begins to get used up. The *USRSPC object name (AEGxxxxLSO, AEGxxxxSYS, AEGxxxxUSR) indicates the type of storage being used. This message (or a series of them) appears in the job log of the process where the storage is being used. This could be either the control region or user shell process. Repeated occurrences of this message as a result of a transaction is an indicator of a possible program loop.

  The LSO storage is a local object space for the control region or shell. The SYS storage is CICS system storage. The USR storage is CICS user storage available to application programs. Refer to the *CICS for iSeries Administration and Operations Guide* if you want to change storage sizes in the SIT.

  Tasks issuing unconditional GETMAIN requests for shared storage are suspended more often as the loop continues and storage is progressively used up. Tasks making storage requests do not need to be in the loop to be affected in this way.
- **Large numbers of autoinitiated tasks**.
- **Large numbers of file accesses** shown for an individual task.

Often loops give some sort of repetitive output. Waits and performance problems never give repetitive output. If the loop produces no output, a repeating pattern can sometimes be obtained by using CICS trace. A procedure for doing this is described in Chapter 7, “Dealing with loops” on page 39.

If you are able to use the CEMT transaction, try issuing the CEMT INQ TASK command repeatedly. If the same transaction is shown to be running each time, this is a further indication that the task is looping. However, note that the CEMT transaction is always running when you use it to inquire on tasks.

If different transactions are seen to be running, this could still indicate a loop, but one that involves more than just a single transaction.

Consider the evidence you have so far. Does it indicate a loop? If so, turn to Chapter 7, “Dealing with loops” on page 39, where there are procedures for defining the limits of the loop.

Poor performance

A performance problem is considered to be one in which system performance is perceptibly degraded, either because tasks fail to start running at all, or because they take a long time to complete once they have started.

In extreme cases, some tasks may be suspended and fail to resume. The problem might then initially be regarded as a wait. Refer to Chapter 6, “Dealing with waits” on page 31.

If CICS is running out of storage, you will see a series of messages prefixed by AEG06xx in the job log of the control region and in the QSYSOPR message queue. The results of this may be the slow running of tasks requiring shared storage.

This may indicate that the system is operating near its maximum capacity, or that a task in error has used up a large amount of shared storage, possibly due to a loop. Use the WRKSYSSTS CL command to determine the amount of storage used.

If there is no such indication, refer to Chapter 8, "Dealing with performance problems" on page 45 for advice on investigating the problem. However, before doing so, be as sure as you can that this is best classified as a performance problem, rather than a wait or a loop.

Poor application design

If you have only a poorly defined set of symptoms that might indicate a loop, or a wait, or possibly a performance problem with an individual transaction, consider the possibility that poor application design might be to blame.

This book does not deal with the principles of application design, or how to check whether poor design is responsible for a problem. See the *CICS for iSeries Application Programming Guide*.

Where to look next

For *transaction abends*, see Chapter 4, "Dealing with transaction abends" on page 25

For *system abends*, see Chapter 5, "Dealing with CICS system abends" on page 29

For *waits*, see Chapter 6, "Dealing with waits" on page 31

For *loops*, see Chapter 7, "Dealing with loops" on page 39

For *poor performance*, see Chapter 8, "Dealing with performance problems" on page 45

For *incorrect output*, see Chapter 9, "Dealing with incorrect output" on page 49

For *storage violations*, see Chapter 10, "Dealing with storage violations" on page 61.

If you have already decided that you should refer the problem to the IBM Support Center, you can find advice on how to go about this in the Basic Operations topic under Systems Management in the iSeries Information Center.

Chapter 3. Sources of information

Begin your problem determination investigation by looking at the following sources of information:

- Your own documentation
- Change log
- Manuals for the products you are using
- Abend codes and error messages
- Dumps
- Transaction inputs and outputs
- Traces.

Your own documentation

"Your own documentation" is the collection of information produced by your organization about what your system and applications do and how they do it. How much of this kind of information you need depends on how familiar you are with the system or application, and could include:

- Program descriptions or functional specifications
- Record layouts and file descriptions
- Flowcharts or other descriptions of the flow of activity in a system
- Statement of inputs and outputs
- Source listings of application programs
- Auxiliary trace information

Change log

The information in your change log can tell you of changes made in the data processing environment that may have caused problems with your application program. To make your change log most useful, include the data concerning hardware changes, system software (such as OS/400 and CICS) changes, application changes, and any modifications made to operating procedures.

Manuals

"Manuals" means the books in the CICS for iSeries and OS/400 libraries, together with in the libraries for any other products you use with your application.

Make sure that the level of any book you refer to matches the level of the software you are using. Problems often arise through using either obsolete information or information about a level of the product that is not yet installed.

For information about working with APARs and reporting system problems, refer to the Basic Operations topic under Systems Management in the Information Center.

Abend codes and error messages

Messages are sent to several destinations, for example:

- CSMT transient data queue for terminal error and abend messages
- Job log for control region messages
- Job log for user shell messages
- QSYSOPR message queue
- QHST history log
- User terminals
- Program message queue

Notes:

1. Abend codes are documented in Appendix B, "Abend codes" on page 93.
2. To increase the amount of message information, issue CHGJOB LOG(4 00 *SECLVL) LOGCLPGM(*YES).
3. First Failure Data Capture (FFDC) information is saved after a system abend.

Dumps

Dumps are an important source of detailed information about problems. Whether they are the result of an abend or of a user request, they allow you to see a snapshot of what was happening in CICS at the moment the dump was taken. Chapter 11, "Using dumps in problem determination" on page 69 contains guidance about using dumps to locate problems in your CICS system. However, because they do provide only a snapshot, you may need to use them in conjunction with other sources of information relating to a longer period of time, such as logs and traces.

Transaction inputs and outputs

Transaction inputs and outputs can be divided into the following areas:

- Terminal data
- Transient data and temporary storage
- Passed information
- Files and databases

Terminal data

Terminal data is very important in solving problems, because it can help you answer the following questions:

- What data did you enter just before the transaction failed?
- Was there any output? If so, what did it look like?

The more you know about the ***input*** at the terminal on which the transaction failed, the better your chance of duplicating the problem in a test environment. However, this information may not be precise, especially if there are many fields on the input screen. You need to provide a quick and easy way for terminal operators to report problems, so that they can report the error while they can still see the data on the screen (or at least remember more clearly what it was).

The ***output*** from a transaction is sometimes easier to capture. If you have a locally attached printer, you can make a copy. (The problem may be that the printer output is incorrect.)

The items to look for on the ***input*** side are:

- Were all necessary input fields entered?
- Were the contents of the input fields correct?
- Which transmit key was used (ENTER or a PF key)?

On the ***output*** screen, check the following points:

- Do all the required fields contain data?
- Is the data correct?
- Is the screen format as it was designed?
- Are there any nondisplay fields used to pass unprotected data?

Transient data and temporary storage

If the program explicitly uses any transient data or temporary storage queues, inspect them to see whether their content is what you expect. You can use the CICS-supplied transaction, CEBR, to inspect transient data queues and temporary storage queues in some detail. See the *CICS for iSeries Application Programming Guide* for information about this transaction.

If you cannot use a CICS/400 screen, you can use the DSPPFM CL command to display the contents of the AAEGxxxxTR and AAEGxxxxTN files.

Even if the program does not use queues, look at the system queues for the CSMT log (or your site replacement) to see whether there are any relevant messages.

What you might want to look for in the queues are:

- Are the required entries there?
- Are the entries in the correct order?
- Is the queue being ***written*** the same one that is being ***read***?

Passed information

Whenever you suspect a problem, check any information that is passed to or from one program to another. The information can be passed in any of the following ways:

- COMMAREA
- Transaction work area (TWA)
- Area obtained by GETMAIN, pointer in TWA
- Through an in-memory table
- Common work area (CWA)
- Terminal user area

Be particularly careful when you are using the common work area (CWA) because you only have one area for an entire CICS system. A transaction may depend on a certain sequence of transactions and some other program may change that sequence. In the CICS/400 environment, if you update the CWA, use the ENQUEUE and DEQUEUE lock mechanism to ensure data integrity. A similar mechanism should be used for tasks updating shared storage by means of GETMAIN commands.

Terminal user areas can have problems because the area is associated with a terminal and not with a particular transaction.

If you are using tables in the CWA, remember that there is no recovery. If a transaction updates the table and then abends, the transaction is backed out but the change is not.

Files and databases

Files and databases are often the main source of transaction input and output; you should always investigate both these areas whenever a program is having problems.

To do this, you need to use the appropriate utilities and diagnostic tools for the data access methods that you have at your installation.

Check the various access paths in files and databases. If you have more than one method of accessing information, one path may be working well while another path may be causing problems.

When looking through the data in files, pay particular attention to the record layout. The program may be using an out-of-date record description.

Traces

CICS provides a tracing facility that enables you to trace transactions. CICS auxiliary trace enables you to write trace records on a sequential device for later formatting and analysis.

For information about the tracing facilities provided by CICS, read Chapter 12, “Using traces in problem determination” on page 73.

Part 2. Dealing with the Problem

Chapter 4. Dealing with transaction abends

This chapter gives guidance about finding the cause of transaction abends. It covers the following subjects:

When a CICS transaction **abends** (ends abnormally) a transaction abend message and an abend code of four alphanumeric characters are sent to the job log and to several other message destinations. This is an example of what the message looks like:

AEG1111 Transaction *transid* **abnormal end** *abcode* in **program** *program name*

The message contains several vital pieces of information. It identifies the transaction (*transid*) that failed, and the program (*program name*) that was being executed when the failure was detected. Most importantly, it gives you the abend code, which indicates the nature of the error.

The transaction abend can originate from several places, and the method you use for problem determination depends on the source of the abend. The procedures are described in the sections that follow. As you go through them, you might like to use the worksheet that is included in Appendix A, "Worksheet for transaction abends" on page 91 to record your findings.

Collecting the evidence

You should find all the evidence you need to investigate the transaction abend in the information sent to the destinations for error messages, and in the transaction dump. The transaction dump is created as a spool file called DPFHDMP with the user data set to the dump code. If no transaction dump has been produced, it is possible that transaction dumping has been suppressed for the transaction (by the transaction definition). For guidance about changing the dumping options so that you get a transaction dump, refer to Chapter 11, "Using dumps in problem determination" on page 69.

Check also to see whether any relevant messages have been sent to message destinations used by CICS to record messages. Look for any messages about files, terminals, or printers that you might be attempting to use. The destinations that you should check for messages include:

- Job logs for both the control region and the user shell
- QSYSOPR message queue
- History log
- CSMT log (if defined)
- EXEC interface block contained in program storage. This may be contained in the transaction dump

What the abend code can tell you

The first thing that the transaction abend code can indicate is whether or not this was a CICS abend. The transaction abend codes of CICS all begin with the letter "A", See Appendix B, "Abend codes" on page 93, and the *CICS for iSeries Application Programming Guide* for a list and descriptions. so those codes beginning with any other letter belong to a user program or to another product. However, be aware that such programs may also use the letter "A" for their abend codes. For the sake of convenience, all such non-CICS abend codes are referred to in this chapter as user abend codes.

If you have received a user abend code, it can still be difficult to find out which program is responsible for it unless you have adequate documentation. For this reason, it is good practice for all programmers who issue abends from within their programs to document the codes in a central location at your installation.

As far as vendor products are concerned, the documentation includes, in most cases, a list of abend codes that are issued from the programs making up the products. This list, together with the documentation for your internal applications, should make it possible for you to find what caused the abend. If it is not clear why the user abend was issued, you might need to describe the problem to the owner of the program.

CICS/400 transaction abend codes

Appendix B, "Abend codes" on page 93 contains a listing of the CICS/400 abend codes. Also refer to the *CICS for iSeries Application Programming Guide* for a description of the EIBRESP field of the EXEC interface block. The list is presented in two sequences: first in EIBRESP code order, and second in abend code order. The condition associated with each abend is also given.

For many abend codes, the cause is simply an unexpected condition on a CICS command, that the program did not handle. The type of condition may help you identify the CICS command causing the abend. If you still cannot find the cause of the problem, continue with the procedure described here.

ASRA and ASRB abends

ASRA and ASRB are system abends but usually indicate a problem in the transaction program.

CICS issues an ASRA abend code when a function check is detected. This can be caused by CICS/400 receiving an unexpected response after requesting an OS/400 function. Abend ASRB is issued if program control detects an error during the processing of an EXEC CICS API request, and in turn issues an ASRB abend for the transaction. The error is related to either internal CICS processing or processing performed by the operating system on behalf of CICS.

To identify further the cause of the abend, use the information sources outlined in "Collecting the evidence" on page 25, together with the methods outlined in the remainder of this chapter, to find out where and why the abend occurred.

Finding where the abend occurred

For transaction dumps, CICS creates a spool file called DPFHDMP. It is created on the output queue for the job where the transaction abend occurred. The dump can be identified by the user data area shown in the WRKSPLF (Work with Spooled Files) CL command. The abend code as identified by CICS appears in this area. If the dump is a system dump, the user data area contains "CICSCRDUMP" to identify it as a control region dump.

Types of transaction dumps

CICS creates two different types of transaction dump. The type created depends on whether a COBOL application program is being executed when the error occurred, or if the error occurred in either an internal CICS function or a supplied transaction.

1. **If a CICS COBOL application is being executed** when an error occurs, CICS creates a COBOL/400 formatted dump. CICS determines whether a COBOL application program is being executed by examining the PPT entry for the current transaction. If the program type is COBOL, a COBOL/400 formatted dump is created. For information about working with COBOL/400 dumps, refer to the *COBOL/400 User's Guide*.
2. **In other cases**, CICS creates a dump consisting of the output of a DMPJOB (Dump Job) CL command. This dump contains basic information about the job that was running the CICS transaction. This dump, along with the job log for the transaction, should provide adequate information for determining the problem.

Dealing with arithmetic exceptions

If the function check was due to an arithmetic error, you need to find the operands used in the last instruction. You need to know the type of arithmetic being done, so that you can tell whether the operands are valid. The interrupt you received tells you what sort of arithmetic the system was doing (binary, packed decimal, or floating point); but you need to determine whether that is what you had intended to do. You might need to consult one of the COBOL language or ILE C language books mentioned in vi.

When you have identified the operands, you need to decide where the problem is. Questions to consider include:

- Has the data been overlaid?
- Is the length field on, for example, an EXEC CICS RECEIVE command, not large enough?
- Has the value been changed by faulty logic?
- Does the data type not match the operation type? For example, if you define the variable as being packed decimal and then you read in binary information, this causes a 'data exception' error.

Analyzing the problem further

You should now know the point in the program at which the abend occurred, and what the program was attempting to do.

- Use OS/400 native facilities, such as the trace facility STRDBG with break points, and the DMPOBJ CL command to dump objects.
- If your program uses or calls other programs or systems, examine the interface and the way you pass data to the program. Are you checking the returned

information from the other system? Incorrect logic paths based on incorrect assumptions can give unpredictable results.

- Examine the flow of your program, using tools like the Execution Diagnostic Facility (CEDF). Check the transient data and temporary storage queues with the CICS browse transaction (CEBR), and use the CICS command-level interpreter (CECI). For information about these transactions, see *CICS for iSeries Application Programming Guide*. If necessary, insert additional statements into the program until you understand the flow. For example, add EXEC CICS WRITEQ, EXEC CICS ENTER TRACENUM, or EXEC CICS ENTER TRACEID statements.
- Look at any trace output you might have, and compare it for differences against the "normal" trace output which is included in the documentation.
- Define the current environment, and try to isolate any changes in it since your program last worked. This can be difficult in large installations, because so many people interact with the systems and slight changes can affect things that seem unconnected.

Chapter 5. Dealing with CICS system abends

The purpose of this chapter is to give guidance about gathering essential information about CICS system abends. If a message has alerted you that a CICS system abend has occurred, you will find this material helpful. This chapter covers the following:

- "The documentation you need"
- "First Failure Data Capture (FFDC) information" on page 30

The message indicating a CICS abend appears in the jog log of the control region and in the QSYSOPR message queue. In addition, other error messages indicating problems with CICS may appear in any of the other CICS message destinations. However, the QSYSOPR message queue contains all the messages pertinent to a CICS system abend.

The cause of a CICS system abend may be related to a problem encountered in a CICS user shell. Use the WRKMSG and DSPLOG CL commands to examine the QSYSOPR and QHST message destinations respectively for CICS messages issued on behalf of user shells at or near the time of the CICS system abend. These messages may provide further clues as to the cause of the abend.

Use the WRKMSGD CL command with the messages indicating the abend. CICS/400 messages are contained within the QCICS/AEGMSG message file. The second-level message text may provide a straightforward explanation of the problem. Checking the QSYSOPR, QHST, or job log messages, to find the second-level text.

If the abend was clearly caused by a storage violation, refer to Chapter 10, "Dealing with storage violations" on page 61. You know when CICS has detected a storage violation, because it issues this message:

AEG0504 Storage control has detected a serious error.

On reading this chapter, you may find that the abend was due to an application error. In this case, you need to examine the application to find out why it caused the abend. However, if it appears that the error is in a CICS module, you need to contact the IBM Support Center. Before doing so, you must gather the First Failure Data Capture (FFDC) information saved by CICS whenever a CICS system abend occurs.

The documentation you need

The primary pieces of documentation you need for investigating abends are as follows:

- The First Failure Data Capture (FFDC) information saved by CICS when a system abend occurs.
- The messages in the QSYSOPR message queue relating to the control region that failed.
- The auxiliary trace objects (if available) for the failed control region.

Whenever CICS invokes its FFDC routines, a series of AEG16xx messages appear in the QSYSOPR message queue. These messages indicate where the problem occurred, and what FFDC captured data relates to the error. FFDC routines cause the logging of a problem on the OS/400 problem log. In addition, the control region may take a CICS system dump. However, this system dump is of little value for problem determination, except by IBM Support Center personnel.

If you had auxiliary trace active at the time of the abend, the auxiliary trace objects should be copied for further examination before starting the control region again. These objects may be of use either to you or to the IBM Support Center in diagnosing the problem. Use the CRTDUPOBJ CL command to copy these objects to a protected object for future use.

If after examining the CICS message destinations for error messages related to the abend, you determine that the error is in a CICS module, the problem should be logged with the IBM Support Center.

First Failure Data Capture (FFDC) information

When a system abend occurs, CICS may have recorded several lines of information relating to the error by First Failure Data Capture (FFDC), saved in the OS/400 problem log. There may be FFDC information for transactions being executed at the time of the abend. In addition, FFDC information may have been captured for the abending control region.

Use the ANZPRB (Analyze Problem) or WRKPRB (Work with Problem) CL commands to examine the problem-related information captured by CICS. The *AS/400 SAA® SystemView® System Manager/400 User's Guide*, SC41-8201-02, gives step-by-step guidance in the use of ANZPRB and WRKPRB.

FFDC information for the CICS abend may contain:

- A symptom string indicating the time of the error. The symptom string contains the CICS function and function statement number from which FFDC was called.
- Control region and user shell job logs.
- Storage space objects for the control region or user shell.
- The internal trace table object for a control region.
- A series of CICS data areas containing specific information related to the error.

The messages within the job logs (either shells or the control region) captured by FFDC may provide clues related to the problem. Examine these messages closely to ensure that the error is indeed a CICS error. If the messages indicates a possible application or CICS system initialization problem, follow the recovery procedures recommended for the message.

However, if you determine that the error is in a CICS module, use the ANZPRB command to log the problem with the IBM Support Center. See the Basic Operations topic under Systems Management in the Information Center for guidance on how to report problems to your IBM Support Center.

Chapter 6. Dealing with waits

This chapter gives you information about what to do if you are aware that a task is in a wait state, or if CICS has stalled during:

- A run
- Initialization
- Termination

This chapter covers the following:

If CICS has stalled, turn directly to "What to do if CICS has stalled" on page 36.

Because CICS uses native services, a wait might be indicated by OS/400, and not CICS. For those problems, refer to the *Work Management* guide.

If you have one or more tasks in a wait state, you should already have carried out preliminary checks to make sure that the problem is best classified as a wait, rather than as a loop or as poor performance. If you have not, you can find guidance about how to do this in Chapter 2, "Classifying the problem" on page 9.

You are unlikely to have direct evidence that a CICS system task is in a wait state, except from a detailed examination of a trace. You are more likely to have noticed that one of your user tasks, or possibly a CICS user task—for example, a CICS-supplied transaction—is waiting. It is possible that a waiting CICS system task could be the cause of the user task having to wait.

For the purpose of this chapter, a task is considered to be in a wait state if it has been suspended after first starting to run. The task is *not* in a wait state if it has been registered to the control region, but has not yet started to run, or if it has been resumed after waiting but cannot, for some reason, start running. These are best regarded as performance problems. See Chapter 8, "Dealing with performance problems" on page 45.

There are two stages in resolving most wait problems involving user tasks. The first stage involves finding out what resource the suspended task is waiting for, and the second stage involves finding out why that resource is not available. This chapter focuses principally on the first of these objectives. However, in some cases there are suggestions of ways in which the constraints on resource availability can be relieved.

If you know that a CICS system task is in a wait state, it does not necessarily indicate an error in CICS. Some system tasks spend long periods in wait states, while they are waiting for work to do.

For further guidance refer to the following:

- If you want guidance about using online or offline techniques to investigate waits, see "Techniques for investigating waits".
- If you know the functional area with which the wait is associated, turn directly to the appropriate section later in this chapter:
  - "Data queue waits (DEQW)" on page 33
  - "Lock waits (LCKW)" on page 35
  - "Message waits (MSGW)" on page 35
  - "Intersystem Communication File waits (ICFW)" on page 35
  - "Display waits (DSPW)" on page 36

Note: Throughout this chapter, the terms "suspension" and "resumption" and "suspended" and "resumed" are used generically. Except where otherwise indicated, they refer to the WAIT and POST processes by which tasks can be made to stop running and then be made ready to run again.

Techniques for investigating waits

You can investigate waits in a CICS system by tracing, or by analysis of the CICS system dump, or by using OS/400 native facilities.

Tracing involves significant processing overhead. However, the information it gives you about system activity in the period leading up to the wait is likely to provide much of the information you need to solve the problem.

If you are able to reproduce the problem, consider using auxiliary tracing.

The OS/400 native facilities for investigating job status can be used to determine the status of a CICS process. STRJOBTRC, ENDJOBTRC, and PRTJOBTRC commands can be used to gather OS/400 trace information. WRKACTJOB and DMPJOB commands can be used to gather job information. In addition, the native debug facilities STRSRVJOB, STRDBG, and ENDSRVJOB commands may be used to determine a current job status.

Using trace

You can find detailed information about the suspension and resumption of tasks during a run of CICS by studying the trace table. Tracing must, of course, be running when the task in question is suspended or resumed, and tracing must be set on.

When you print the contents of a trace table, or look at a print of the trace table, you can find trace entries relating to a particular task using the task number. Trace entries can be selected using the task number, and individual trace entries each contain the task number. Note that task numbers may not be unique. Within a single user shell, only the last digit of the task number changes. If more than 10 tasks have been run since the user shell was initialized, trace entries for a single task number may relate to two different tasks. In this situation, the date and time fields of each trace entry should be used to identify a particular task or tasks.

For general guidance about setting tracing options and interpreting trace entries, see Chapter 12, "Using traces in problem determination" on page 73.

The formatted CICS system dump

If you are suitably authorized, you can request a CICS system dump using the CEMT PERFORM SNAP command. Make sure that the task in question is waiting when you take the dump, so that you capture information that is relevant to the wait.

For information about reading dumps, see "Looking at dumps" on page 72.

Using the WRKACTJOB command

The WRKACTJOB command is the most useful command for gathering job-related information. If you know the full OS/400 job name for the control region job and understand that the STRCICSUSR command identifies a CICS user shell job, you can identify each control region and CICS user shell job active in the system. An OS/400 job may be in a wait state for many different reasons. The job status portion of the WRKACTJOB display gives an indication if the job is in a wait state.

If the job status does not change from a wait state after refreshing the status display a few times, you may need to determine what the job is waiting on. Follow the steps indicated in *Work Management* to determine the actions required for each type of wait state.

Of particular concern to CICS user shell jobs are the following wait states:

MSGW Message wait

DEQW Data queue wait

DSPW Display wait

ICFW Intersystem communication file wait

LCKW Lock wait

Note: The most common wait state encountered by CICS shell and control region jobs is the DEQW.

Data queue waits (DEQW)

Data queue waits may originate out of the control region when it is waiting for work, or for numerous reasons listed below if a CICS shell job is running.

The control region

When the control region has been initialized, the normal job status for the control region is a DEQW. CICS services requiring control region action communicate with the control region through the data queuing facility. If there is no work for the control region to process, it remains in a DEQW job status.

The application shells

CICS/400 uses the data queuing facilities of the OS/400 to communicate between the control region and user shell jobs. In addition, CICS/400 uses the data queuing facility for suspending or resuming tasks requiring resources that are not available at the time of the request. This data queuing facility is used to control the task suspend or resume processing for those EXEC CICS commands with a WAIT option, or which cause a wait to occur.

In CICS/400, waits may occur in the following functional areas:

Interval control	For RETRIEVE with WAIT, WAIT EVENT, DELAY, and CANCEL requests.
Journal control	When a WRITE JOURNALNUM causes a journal switch, the task issuing the request waits until the file switch takes place.
Terminal control	For CONVERSE and RECEIVE commands.
Transient data	All extrapartition requests are processed through the control region. The task issuing the extrapartition request waits until the request is processed by the control region.
Temporary storage	WRITE requests to main storage when shared storage is unavailable. The task making the request is suspended until enough shared storage is available to process the request.
Storage control	GETMAIN requests are issued for shared storage and not enough shared storage is available to process the requests.
System information	For the SET FILE CLOSED with WAIT command. The task issuing the request is suspended until an indication is received that the file has been closed or the request failed.

Identifying DEQW within a CICS trace

It is easy to identify the type of DEQW using a CICS formatted trace. If a CICS user shell is in a DEQW state and auxiliary trace is active, format the active trace object for the job number in question. The last formatted trace entry for the suspected job should be an EVENT trace entry from the CICS module AEGTCPQD. This is the CICS module that processes data queuing requests for all CICS components. Look backward within the formatted trace entries until you find an ENTRY trace record belonging to one of the components referred to previously. The function code in the trace should correspond to the EIBFN for the EXEC CICS command being executed; this should indicate the cause for the DEQW.

If the entire trace file is formatted, the AEGTCPQD entry is followed by a number of control region trace entries (identified by a task number of all zeros). When the data queuing request has been processed by the control region, the task trace contains another AEGTCPDQ EVENT trace followed by an EXIT trace, followed eventually by an EXIT trace entry from the component program processing the initial EXEC CICS command.

Storage control and main temporary storage requests causing a task wait are preceded by messages written to the QSYSOPR message queue and to the control region job log, indicating that the system storage area has been extended a number of times and that no more extents remain. If shared storage is never freed with a FREEMAIN request, these tasks will wait forever, or until the control region is terminated. See “Is the system short on storage?” on page 37 for further information about shared storage problems.

Lock waits (LCKW)

CICS processes using the OS/400 data management facilities may encounter lock waits. For example, if two CICS tasks require the same record for update on the same file, the OS/400 data management facilities force one of the tasks to wait until the lock is freed. File control, temporary storage, and transient data requests for recoverable resources may encounter lock waits. Lock waits on system objects can also occur, if the system is being run continually at full capacity.

Use the WRKACTJOB command to work with the job in a lock wait. The "work with job lock" option shows all the locks and lock types held by a particular job. This should indicate any file locks being held by a job. By using the WRKOBJLCK (work object locks) command on the locked files, you can determine which job is holding the file lock. Your CICS job may be waiting on this file lock.

Task control uses its own locking mechanism to control the EXEC CICS ENQ and DEQ processing. EXEC CICS ENQ requests against the same resource cause a resource wait; these appear as LCKW waits on a WRKACTJOB display. However, the work with locks display does not indicate any object locks being held.

To identify a task control ENQ wait, a CICS formatted trace is necessary for the suspected task. If the last entry from the formatted trace output is an EVENT trace from AEGKCENQ, the task is in an ENQ wait, waiting for another CICS task DEQ on a resource. The ENQ RESOURCE appears in the message data of the EVENT trace entry recorded by AEGKCENQ.

Message waits (MSGW)

During control region initialization, a number of errors may be encountered by a component. For example, journal control may find that a file object referred to by a journal control table (JCT) entry does not exist. Each initial component may issue a message or a series of messages indicating problems during initialization.

The control region regains control after a component's initialization is complete. If initialization errors are detected, the control region issues message AEG1530 asking the operator whether the control region startup should continue. This message requires a reply, which puts the control region into a MSGW job status. When a reply is given for the message, the control region status changes to RUN until either initialization completes or another error is encountered.

In addition, transaction processing may result in an OS/400 message being sent to the job, requiring a response. This would put the job into a message wait state. Use the WRKMSGD command to determine the proper response to the message displayed so that processing can continue or be ended.

Intersystem Communication File waits (ICFW)

CICS requests for remote resources, or APPC requests, may encounter ICF waits while processing the request. An ICF wait indicates that the job is waiting for the completion of an input/output operation to the CICS/400 Intersystem Communication function file.

Display waits (DSPW)

Jobs related to CICS only go into display waits if some type of debugging (either CEDF or STRDBG) activity is taking place against the job requiring input from the display to continue. When a response has been made to the debugging request, the display wait should be satisfied and the job should continue in RUN status.

What to do if CICS has stalled

CICS may stall during initialization, when it is running apparently "normally", or during shutdown.

These possibilities are dealt with separately in:

- "CICS has stalled during initialization"
- "CICS has stalled during a run"
- "CICS has stalled during shutdown" on page 37

CICS has stalled during initialization

If CICS stalls during initialization (on cold start, warm start, or emergency restart), the first place to look is the QSYSOPR message queue on the job log. This tells you how far initialization has progressed.

Note that there might be significant delays at specific stages of initialization, depending on how CICS was last shut down.

On cold start, loading the GRPLIST definitions can take several minutes. For each initialization stage, CICS issues a message indicating which CICS resource tables are being installed and initialized.

If you find that unexpected delays occur during CICS initialization, consider the messages that have already been sent to the QSYSOPR message queue and the job log, and see if they suggest the reason for the wait.

CICS has stalled during a run

If a CICS region that has been running normally stalls, so that it produces no output and accepts no input, the scope of the problem is potentially system-wide. The problem might be confined exclusively to CICS, or it could be caused by another process running under the OS/400.

Look first on your QSYSOPR message queue for any messages. Look particularly for messages indicating that operator intervention is needed; for example, to change a tape volume. The action could be required on behalf of a CICS task, or it could be for any other program with which CICS needs to interface.

If there is no operator action outstanding, use the WRKACTJOB CL command to see what the processor usage is for CICS. If you find the value is very high, this probably indicates that a task is looping.

If the processor usage is low, CICS is doing very little work. Some of the possible reasons are:

- The system definition parameters are not suitable for your system.
- The system is short on storage, and new tasks cannot be started. This situation is unlikely to last for long unless old tasks cannot, for some reason, be purged.

- There is an exclusive control conflict for a disk file or tape.
- There is a problem with the communication access method.
- There is a CICS system error.

Some questions that you need to consider are discussed under the headings that follow. For some of the investigations, you might need to refer to a system dump of the CICS control region. Make sure that CICS *is* apparently stalled at the time you take the dump, because otherwise it won't provide the evidence you need.

Is the OS/400 job description correct?

The OS/400 job description under which the CICS job is running is incorrectly defined.

Is the system short on storage?

If the system is short on storage, a message is sent to the control region job log.

CICS is unlikely to stall for reasons of being short on storage, because tasks are normally purged to relieve the condition whenever it is detected. However, for a task to be stall purgeable, PURGE(*YES) must be specified in the installed transaction definition (PCT), and WAITTIME must be set to a deadlock time-out value. Generally, the shorter the value of WAITTIME, the greater the likelihood that the task will be purged when CICS is short on storage.

Note that the default values are PURGE(*YES) and WAITTIME(0), that is, tasks are, by default, purgeable.

Is there a CICS system error?

If you have investigated all the task activity, and all the other possibilities from the list, and you have still not found an explanation for the stall, it is possible that there is a CICS system error. Contact your IBM Support Center with the problem.

CICS has stalled during shutdown

Waits often occur when CICS is being quiesced because some terminal input or output has not been completed. To test this possibility, try using the CEMT transaction to inquire on the tasks currently in the system.

CICS control region shutdown begins when any of the following commands is issued:

- ENDJOB CL command
- ENDSBS CL command
- ENDCICS CL command

If you cannot use the CEMT transaction, use the native OS/400 facilities. For example, the WRKACTJOB CL command can be used to discover the CICS wait state.

The action to take next depends on whether or not there are current user tasks. It is based on your being able to use the CEMT transaction.

- **If there are user tasks currently in the system**:

  Check what they are. A task may be performing a prolonged termination routine, or it might be waiting on a resource before it can complete its processing. It is also possible that a task is waiting for operator intervention.

Determine what type of terminal is associated with the task. If the terminal is a display device, some keyboard input might be expected. If it is a printer, it might have been powered off, or it might have run out of paper.

- **If there are no user tasks in the system**:

  It may be that one or more terminals have not been closed. Use the CEMT transaction to see which terminals are currently INSERVICE, and then use the CEMT SET command to place them OUTSERVICE.

Chapter 7. Dealing with loops

This chapter outlines procedures for finding which program is involved in a loop that does not end. This chapter covers:

- "Tight loops and nonyielding loops"
- "Yielding loops" on page 40
- "Investigating loops" on page 41
- "What to do if you cannot find the reason for a loop" on page 43.

A loop is a sequence of instructions that is executed repetitively. Loops that are coded into applications must always be guaranteed to terminate, because otherwise you could get any of the symptoms of loops described in Chapter 2, "Classifying the problem" on page 9.

If a loop does not terminate, it could be that the termination condition can never occur, or it might not be tested for, or the conditional branch could erroneously cause the loop to be executed over again when the condition is met.

If you find that the looping code is in one of your applications, you need to check through the code to find out which instructions are in error.

Tight loops and nonyielding loops

Figure 1 gives a specific example of code containing a simple tight loop.

```
PROCEDURE DIVISION.
    EXEC CICS
        HANDLE CONDITION ERROR(ERROR-EXIT)
              ENDFILE(END-MSG)
    END-EXEC
SET-TIME.
    EXEC CICS
        ASKTIME
        ABSTIME(TIME)
    END-EXEC
 NEW-LINE-ATTRIBUTE.
    GO TO NEW-LINE-ATTRIBUTE
    MOVE LOW-VALUES TO PRNTAREA
    MOVE DFHBMPNL TO PRNTAREA
```

Figure 1. Example of COBOL code containing a tight loop

Tight loops and nonyielding loops are characterized by the fact that the looping task can never be suspended within the limits of the loop.

A tight loop is one involving a single program, where the same instructions are executed repeatedly and control is never returned to CICS/400.

Figure 2 shows a tight loop, with an indefinite number of instructions contained in the loop. None of the instructions returns control to CICS/400. In the extreme case, there could be a single instruction in the loop, causing a branch to itself, as in Figure 1.

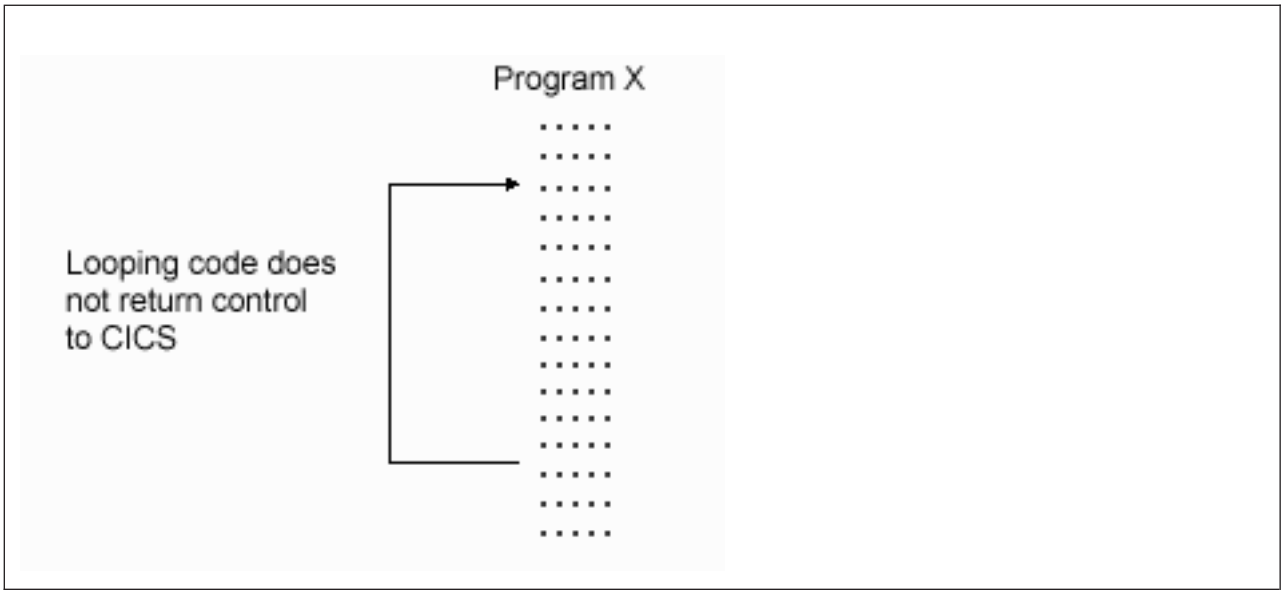


Figure 2. A tight loop

A nonyielding loop is also contained in a single program, but it differs from a tight loop in that control is returned temporarily from the program to CICS. That is, it contains at least one EXEC CICS command within the loop.

Figure 3 shows a nonyielding loop.

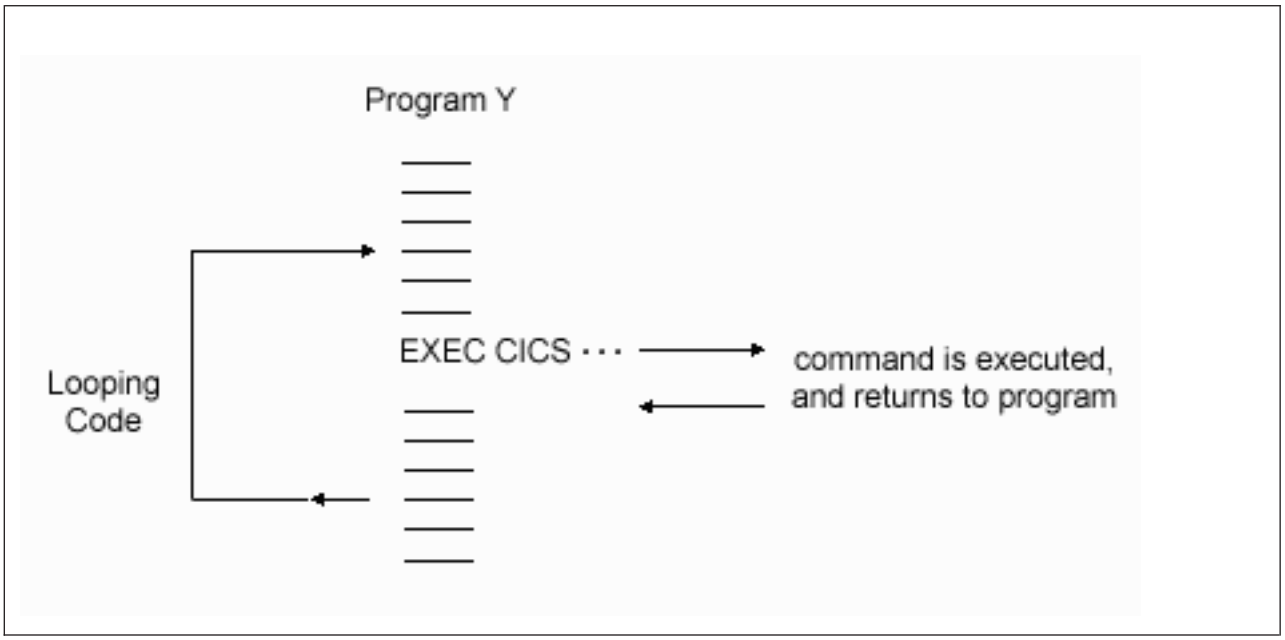


Figure 3. A nonyielding loop

Yielding loops

Yielding loops are characterized by returning control at some point to a CICS/400 routine that can suspend the looping task. However, the looping task is eventually resumed, and so the loop continues.

Yielding loops typically involve a number of programs. The programs might be linked to and returned from, or control might be transferred from one program to another in the loop. A yielding loop can also be confined to just one program, in which case it must contain at least one wait-enabling CICS/400 command.

Figure 4 shows a yielding loop involving two programs, A and B. Program A links to program B, and then program B subsequently returns.
You can detect a yielding loop only by circumstantial evidence such as repetitive

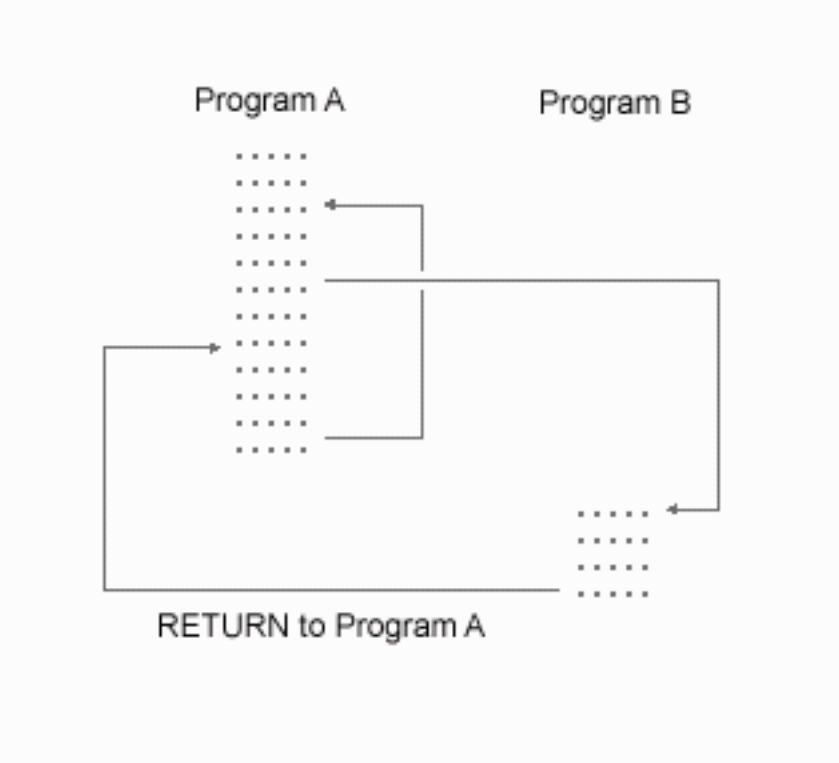


Figure 4. A yielding loop

output, or excessive use of storage. A fuller description of what to look out for is given in "Loops" on page 15.

If you suspect that you have a yielding loop, turn to "Investigating loops" for further guidance.

Investigating loops

You probably suspect that you have a loop through circumstantial evidence. You might, for example, have seen some sort of repetitive output, or statistics might have shown an excessive number of I/O operations or requests for storage. These types of symptom can indicate that your code contains a yielding loop.

The nature of the symptoms may indicate which transaction is involved, but you probably need to use trace to define the limits of the loop.

Use auxiliary trace to capture the trace entries, to ensure that the entire loop is captured in the trace data. If you use internal trace, there is a danger that wraparound will prevent you from seeing the whole loop.

Be sure that the control region executing the transaction has auxiliary trace capability. Activate auxiliary trace using the CEMT TRACEDEST AUXSTATUS(AUXSTART) command, if necessary. This should capture the trace data of the looping task.

When you have captured the trace data, you need to purge the looping task from the system. Use the CEMT INQ TASK command to find the number of the task, and then purge it using the CEMT SET TASK PURGE command. This causes the transaction to abend, and to produce a transaction dump of the task storage areas.

The documentation you need

When investigating loops, you should consider the following information:

- Compiled program listing of the suspected program
- Formatted CICS trace output
- CICS transaction dump for the suspected program.
- OS/400 trace output for the looping job.
- WRKACTJOB display showing the program stack at the time of the suspected loop. This may show the statements of the program suspected of looping.

The trace data and the program listings should enable you to identify the limits of the loop. You need the transaction dump to examine the user storage for the program. The data you find there could provide the evidence you need to explain why the loop occurred.

Identifying the loop

There are two approaches to identifying loops in user programs. You can use the trace table, or you can look in the transaction dump.

Using the trace table

Assuming that CICS auxiliary trace was active and that the trace data has been formatted using the PRTCICSTRC command, go to the last entry in the formatted trace output and work backward. If the looping task was canceled, a trace entry should appear indicating the abend code.

Before the abend entry, look for repetitive trace entries from program AEGEIEIP—the EXEC Interface program. There should be matching ENTRY and EXIT trace entries for each EXEC CICS command executed from the application program. The function corresponds to the value of the EIBFN corresponding to the EXEC CICS command being executed. Refer to the *CICS for iSeries Application Programming Guide* for a list of these values. On each EXIT trace entry from AEGEIEIP, the return code value corresponds to the EIBRESP—the response code from the request.

If you see a repeating pattern of AEGEIEIP ENTRY and EXIT trace entries, you should be able to match these entries to statements in your source code. This should define the limits of your loop. If you do not see such a repeating pattern, it is likely that the program has a tight loop, where no EXEC CICS commands are being executed.

Using the COBOL transaction dump

The COBOL/400 formatted dump created when the transaction abends provides information indicating the state of the program at the time of the dump. This pinpoints the statement being executed at the time of the dump. Because CICS

invokes the COBOL dump, the statement being executed is normally a call to AEGEIEIP. However, this statement number should indicate a point in the application being executed as part of the loop.

When you have located a point within the loop, work through the source code and try to find the limits of the loop.

Finding the reason for the loop

Look carefully at the statements contained in the loop. Does the logic of the code suggest why the loop occurred? If not, you need to examine the contents of data fields in the task user storage. Look particularly for unexpected response codes, and null values when finite values are expected. Programs can react unpredictably when they encounter these conditions, unless they are tested for and handled accordingly.

What to do if you cannot find the reason for a loop

If you cannot find the reason for a nonyielding or a yielding loop using the techniques outlined above, there are three more approaches that you can adopt:

- Use an interactive runtime debugging tool.
- Use the interactive tools that CICS provides.
- Modify the program, and run it again.

Investigating loops using interactive tools

The OS/400 interactive debugging facilities (STRDBG) can be used to debug non-CICS parts of the COBOL application program. Refer to the *COBOL/400 User's Guide* for more information about debugging COBOL/400 programs.

For information about debugging C programs, refer to *ILE Concepts*.

CICS-supplied debugging facilities

If your program contains a loop, you can use the execution diagnostic facility (CEDF) to look at the various parts of your program and storage at each interaction with CICS/400. If you suspect that some unexpected return code might have caused the problem, CEDF is a convenient way of investigating the possibility.

The CEBR transaction is also useful for investigating loops. You can use CEBR to examine the status of queues during the execution of your program. You can use the command-level interpreter CECI to examine the status of the files required by your applications. Programs can react unpredictably if records and queue entries are not found, and the conditions are not handled.

Further information on using debugging tools, the CEDF, CECI, and CEBR transactions, and on handling program exception conditions can be found in *CICS for iSeries Application Programming Guide*

Modifying your program to investigate the loop

If the program is extremely complex, or the data path difficult to follow, you may need to insert additional statements into the source code. Even extra EXEC CICS ASKTIME commands allow you to use CEDF and inspect the program at more points. You can also request dumps from within your program, and insert user trace entries, to help you find the reason for the loop.

Chapter 8. Dealing with performance problems

This chapter covers the following:

If you are only aware that performance is poor, and you have not yet found which of these is relevant to your system, read "Finding the bottleneck". When you have a performance problem, you might find that it is characterized by a particular processing bottleneck. If so, refer directly to the relevant section:

For general advice on designing applications that run efficiently, see the *CICS for iSeries Application Programming Guide*.

Finding the bottleneck

Each bottleneck is affected by a different set of system parameters, and you may find that adjusting the parameters solves the problem. You need to determine which bottleneck is causing your performance problem, so you can determine which parameters you need to change.

If performance is particularly poor for any of the tasks in your CICS system, you might be able to capture useful information about them with the WRKACTJOB command. However, tasks usually run more quickly than you can inquire on them, even though there may still be a performance problem. You then need to consider using tracing to get the information you need.

Initial start of task: If a task has not started, you cannot get any information about its status online. The CEMT INQ TASK command returns a response indicating that the task is not known. If the task has not already run and ended, this response means that it has not started.

Guidance about finding out why tasks take a long time to get started is given in "Why tasks fail to get started" on page 46.

Task begins but does not continue: A task may start, but then take a long time to continue running. In this case, the CEMT INQ TASK command returns a status of 'READY' for the task. If you keep getting this response and the task fails to do anything, it is likely that the task you are inquiring on is not getting its control region registration.

The delay might be too short for you to use the CEMT INQ TASK command in this way, but still long enough to cause a performance problem. In this case, you need to use tracing for the task, to tell you how long the task had to wait for control region registration.

The execute, suspend, and resume cycle: If performance is poor and tasks are getting attached and executed, the problem lies with the execute, suspend, and resume cycle. Tasks run to completion, but the overall performance is poor. If you

are able to determine that tasks are getting attached and then registered, read "Why tasks take a long time to complete" on page 47.

Why tasks fail to get started

A task might fail to get started for one of the following reasons:

- The interval specified on an EXEC CICS START command might not have expired, or the time specified might not have been reached, or there might be some error affecting interval control. You need to consider this only if INTERVAL or TIME was specified on the START command.
- The terminal specified on an START command might not be available. It could be currently OUTSERVICE, or executing some other task. You can check its status using the CEMT INQ TERMINAL command, and perhaps take some remedial action.

  Remember that several tasks might be queued on the terminal, some of which might require operator interaction. In this case, the transaction to be started might not get attached for a considerable time.
- A remote system specified on an START command might not be available. Or an error condition might have been detected in the remote system; this error would not be reported back to the local system. You can use the CEMT INQ TERMINAL command to inquire on the status of the remote system.
- Batch STRCICSUSR requests may not start due to the system being overloaded. Or the job class for the request is already full, and the job is awaiting scheduling. Using the work management commands, WRKACTJOB, WRKJOBQ, and WRKSYSSTS, may give you an indication as to the status of the jobs on the system.
- The control region for the requested STRCICSUSR may not be active. This causes an error message to appear in the job log of the requester.
- The system may be overloaded. You can use the work management commands to track the reason for the overload, and to make system changes to allow the tasks to start.

Why tasks fail to get attached to the control region

There are several reasons why a task may not attach to the control region. The most basic reason is that the system is overloaded, and the control region is running at a low priority compared to the rest of the system. In this situation, the control region is not getting the cycles needed to process the work requested.

Another reason may be a lack of shared storage in the control region to process the request. If the control region job log has a number of messages indicating that the shared storage space object is being extended a number of times, this could mean that the control region needs more shared storage space to process all the activity.

Using trace: You can use trace if you want to find out just how long an individual task waits to be attached to the control region. When you have selected the options, start tracing to the auxiliary trace and attempt to initiate the task. Format the trace using the PRTCICSTRC CL command.

Look for the EVENT trace entry identified by AEGTCPQD, and the job number and the task number of the user shell process. Following this, there should be a number of trace entries with a task number of all zeros and the job number of the control region, ending with an EXIT trace entry from AEGTCPQD. If the last trace

entry is the user shell trace entry for AEGTCPQD, the task has not been attached to the control region. In a WRKACTJOB display, the OS/400 job would be in a dequeue wait (DEQW) status.

The elapsed time between the AEGTCPQD EVENT trace entry of the user shell and the AEGTCPQD EXIT trace entry from the control region is the time taken for the task to become attached to the control region.

Using work management CL commands: You can use these commands to give you clues about problems with any CICS-related jobs. By adjusting the job priority, run priority, and other scheduling-related areas, the CICS-related performance problem may be overcome.

Why tasks take a long time to complete

When a ready task is started, it becomes a running task. It is unlikely to complete without being interrupted at least once, and it is likely to be interrupted many times during its lifetime.

The longer the task spends in the nonrunning state, the greater your perception of performance degradation. In extreme cases, the task might spend so long in the nonrunning state that it is apparently waiting indefinitely. It is not likely to remain 'ready' indefinitely without running, but it could spend so long suspended that you would probably classify the problem as a wait. If you feel that your problem is a wait, see Chapter 6, "Dealing with waits" on page 31.

Here are some factors that can affect how long tasks take to complete:

- System loading
- Time-out interval for tasks
- Use of remote resources
- Use of shared storage

Each of these factors is considered in turn.

The effect of system loading on performance

The greatest factor affecting the time taken for a task to complete is system loading. Note in particular that there is a critical loading beyond which performance is degraded severely for only a small increase in transaction throughput.

The effect of task time-out interval on performance

The time-out interval is the length of time a task can wait on a resource before it is removed from the suspended state. A transaction that times out is normally abended.

Any task in the system can use resources in an exclusive manner and, therefore, not allow other tasks access to those resources. Normally, a task with a large time-out interval is likely to hold onto resources longer than a task with a short time-out interval. Such a task has a greater chance of preventing other tasks from running. It follows that task time-out intervals should be chosen with care, to optimize the use of resources by all the tasks that need them.

Use of remote resources

CICS tasks requiring remote resources may take longer than a similar task using local resources. It takes longer to allocate the APPC session to the remote system and schedule the work in the remote system. CICS/400 depends on the job scheduling of the OS/400. If a task requiring remote resources takes a long time, check how the job is scheduled and the priority of the job processing the remote request.

Use of shared storage

If your applications perform many GETMAIN requests for shared storage, without a corresponding number of FREEMAIN requests, tasks may wait for shared storage to become available for use. If this is the case, make adjustments to the application to lessen the length of time shared storage is required, or increase the shared storage size of the control region.

Chapter 9. Dealing with incorrect output

The term “incorrect output” can be interpreted in many different ways, and its meaning for the purpose of problem determination is explained in Chapter 2, “Classifying the problem” on page 9.

The various categories of incorrect output are dealt with in this chapter:

- “Wrong data has been displayed on a terminal”
- “Incorrect data is present on a physical file” on page 52
- “An application did not work as expected” on page 52
- “Your transaction produced no output at all” on page 53
- “Your transaction produced wrong output” on page 56

Wrong data has been displayed on a terminal

There are many reasons why you might get the wrong data displayed; some are system-related causes and some are application-related causes. If you think that it is system-related, read this section for some suggestions on likely areas in which to start your investigations.

For the present purpose, a terminal is considered to be any device where data can be displayed. It might be some unit with a screen, or it could be a printer. Many other types of terminals are recognized by CICS, including remote CICS systems or batch processes; but they are not considered in this chapter on incorrect output.

Broadly, there are two types of incorrect output that might be displayed:

- The data or information is wrong, so unexpected values appear
- The layout is incorrect, that is, the data is formatted wrongly.

In practice, you may sometimes find it difficult to distinguish between incorrect data and incorrect formatting. In fact, you seldom need to make this classification when you are debugging this type of problem.

Sometimes, you might find that a transaction runs satisfactorily at one terminal, but fails to give the correct output on another. This is probably due to the different characteristics of the different terminals; you should find the answer to the problem in the sections that follow.

The preliminary information you need to get

Before you can investigate the reasons why incorrect output is displayed at a terminal, you need to gather some information about the transaction running at the terminal, and about the terminal itself.

The first thing you need to know is the identity of the transaction associated with the incorrect output. Also, for an autoinstalled terminal, you need to know the DEVTYPE parameter value, to ensure that you inquire on the correct terminal type.

Depending on the symptoms you have experienced, you need to examine the definitions for the transaction, and for the affected terminal. The attributes most likely to be of interest are SCRNSZE (on the ADDCICSPCT command) and

ALTSCN (on the ADDCICSTCT command). The resource definition CL commands are described in the *CICS for iSeries Administration and Operations Guide*. If these options conflict with each other, the results received on the terminal may be incorrect.

Specific types of incorrect output, and their possible causes

This section contains some suggestions about what to do for specific types of incorrect output, and what might be at fault.

Logon rejection message

If you get a logon rejection message when you attempt to log on to CICS, it could be that the TCT definitions for the terminal are incorrect. A message recording the failure is written to the job log and the CSMT log.

If you have a persistent problem with logon rejection, you should check the CICS/400 message destinations for error messages.

Unexpected messages

Messages that are prefixed by "AEG" originate from CICS. To get help for the message, place the cursor on the messages and press F1. If you cannot access a CICS/400 user shell for any reason, use the DSPMSGD (Display Message Description) command to view second-level text for more information related to the error.

The keywords referred to in the following examples are keywords used on the ADDCICSTCT, CHGCICSTCT, and DSPCICSTCT CL commands. The following are examples of common errors that can cause messages to be displayed:

- ALTSCN(*YES) has been specified and too many rows have been specified for ALTSCN in the definition for the terminal.
- An application has sent a spurious hexadecimal value corresponding to a control character in a data stream. For example, X'11' is understood as "set buffer address" by a 3270 terminal, and the values that follow are interpreted as the new buffer address. This eventually causes a TERMERR response and possibly an ATNI abend.

If you suspect this may be the cause of the problem, check your application code carefully to make sure it does not send any unintended control characters.

Unexpected appearance of uppercase or lowercase characters

If the data displayed on your terminal has unexpectedly been translated into uppercase characters, or if you have some lowercase characters when you were expecting uppercase translation, you need to look at the options governing the translation.

The following are the significant properties of the various translation options you have:

- The ASIS option for BMS or terminal control specifies that lowercase characters in an input data stream are *not* to be translated to uppercase.

  ASIS overrides the value in the upper case translation field of the TCT definition.
- The upper case translation attribute of the terminal definition states whether lowercase characters in input data streams are to be translated to uppercase for terminals with this definition.

  ASIS overrides the value in the upper case translation field of the TCT definition.

- If the ASIS option is NOT specified, and if the upper case translate attribute of the terminal definitions specifies *YES, the data presented to the transaction *is* translated.

Initiating a CRTE session

The input required to start a CRTE routing session is of the form:

```
CRTE SYSID(xxxx)
```

Translation to uppercase is dictated by the TCT entry of the terminal at which the CRTE command was entered.

Input within the CRTE session

During the CRTE routing session, uppercase translation is dictated by the type of terminal at which the CRTE command was initiated.

If the first six characters entered on a screen are `CANCEL`, uppercase translation will take place.

Katakana terminals – mixed English and Katakana characters

If you are using a Katakana terminal, you might see some messages containing mixed English and Katakana characters. That is because Katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as Katakana characters. If you have any Katakana terminals connected to your CICS system, specify KATAKANA(*YES) and UCTRN(*YES) in the TCT for the terminal to ensure that messages contain uppercase characters only.

Wrong data values are displayed

If the data values are wrong on the user's part of the screen (the space above the operator information area), or in the hardcopy produced by a printer, it is likely that the application is at fault.

Some data is not displayed

If you find that some data is not being displayed, consider the following possibilities:

- SCRNSZE(*ALT) might be specified in the PCT definition for the transaction running at the terminal, and the default value for ALTSCN is allowed in the TCT definition.

  The default value is ALTSCN(*NONE) so no data could be displayed if SCRNSZE(*ALT) was specified.

The data is formatted incorrectly

Incorrect formatting of data can have a wide range of causes, but here are some suggestions of areas that can sometimes be troublesome:

- BMS maps are incorrect.
- Applications have not been recompiled with the latest maps.
- For a screen display, the number of columns specified must be less than or equal to the line width. For a printer, the number of columns specified must be *less than* the line width, or else both BMS (if you are using it) and the printer might provide a new line; and you will get extra spacing that you don't want.
- If you get extra line feeds and form feeds on your printer, it could be that an application is sending control characters that are not required because the printer is already providing end-of-line and end-of-form operations.

  If your application is handling the buffering of output to a printer, make sure that an "end of message" control character is sent at the end of every buffer full of data. Otherwise, the printer might put the next data it receives on a new line.

Incorrect data is present on a physical file

An error can occasionally occur because OS/400 allows a record to be read by one transaction while another transaction is updating it.

If the first transaction were to take some action based on the value of the record, the action would probably be erroneous.

For example, in inventory control, a warehouse has 150 items in stock. One hundred items are sold to a customer, who is promised delivery within 24 hours. The invoice is prepared, and this causes a transaction to be invoked that is designed to read the inventory record from a physical file and update it accordingly.

In the meantime, a second customer also asks for 100 items. The salesperson uses a terminal to inquire on the number currently in stock. The "inquire" transaction reads the record that has been read for update but not yet rewritten, and returns the information that there are 150 items. This customer, too, is promised delivery within 24 hours.

Errors of this kind are very difficult to guard against, but be aware that they can happen.

An application did not work as expected

It is not possible to give specific advice on dealing with this sort of problem, but the points and techniques that follow should help you to find the area where the failure is occurring.

General points for you to consider

1. Make sure you can define exactly what happened, and how this differs from what you expected to happen.
2. Check the commands you are using for accuracy and completeness. For programming information, see the *CICS for iSeries Application Programming Guide*. If you used default values, are these the values you really want? Does the description of the effect of each command match your expectations?
3. Can you identify a failing sequence of commands? If so, can it be reproduced using the command-level interpreter CECI?
4. Consider the resources required by the application. Are they defined as expected?
5. For "input" type requests, does the item exist? You can verify this using offline utilities.
6. For "output" type requests, is the item created? Verify that the before- and after-images are as expected.

Using traces and dumps

Traces and dumps can give you valuable information about unusual conditions that might be causing your application to work in an unexpected way.

1. If the path through the transaction is indeterminate, insert user trace entries at all the principal points.
2. If you know the point in the code where the error occurs, insert a CICS dump request immediately after it, using the EXEC CICS DUMP command.

3. Run the transaction after activating auxiliary trace options, and wait until the dump request is executed. Format the auxiliary trace table and examine the trace entries before the failure. Look in particular for unusual or unexpected conditions, possibly ones that the application is not designed to handle.

Your transaction produced no output at all

If your transaction produced no output at all, you need to carry out some preliminary checks before looking at the problem in detail. You might be able to find a simple explanation for the failure.

Are there any messages explaining why there is no output?

Look carefully in each of the CICS/400 message destinations for any messages that might relate to the transaction. You might find one there that explains why you received no output.

If you can find no such message, the next step is to get some information about the status of the transaction that produced no output, your terminal, and the CICS system.

Can you use the terminal where the transaction should have started?

Go to the terminal where the transaction should have started, and note whether the keyboard is locked. If it is, reset the terminal. Now try issuing the CEMT INQ TASK command (or your site replacement) from the terminal.

If you can't issue the CEMT INQ TASK command from the terminal, one of the following explanations applies:

- The task that produced no output is still attached to the terminal
- The terminal where you made the inquiry is not in service
- There is a system-wide problem
- You are not authorized to use the CEMT transaction

Try to find a terminal where you can issue the CEMT INQ TASK command. If no terminal seems to work, there is probably a system-wide problem. Otherwise, see if the task you are investigating is shown in the summary.

- If the task is shown, it is probably still attached, and either looping or waiting. Refer to "What to do if the task is still in the system" on page 54 to see what to do next.
- If the task is not shown, there is a problem with the terminal where you first attempted to issue the CEMT INQ TASK command.

If you are able to issue the CEMT INQ TASK command from the terminal where the transaction was attached, one of the following explanations applies:

- The transaction gave no output because it never started.
- The transaction ran without producing any output, and terminated.
- The transaction started at another terminal, and might still be in the system. If it is still in the system, you can see it in the task summary that returned by the CEMT INQ TASK command. It is probably looping or waiting.

  Refer to "What to do if the task is still in the system" on page 54 for advice about what to do next. If you don't see the task in the summary, refer to "What to do if the task is not in the system" on page 54.

What to do if the task is still in the system

If you obtained no output and the task is still in the system, it is either waiting for a resource, or looping. You should get an indication of which of these two conditions is the most likely from the status for the task returned by the CEMT INQ TASK command.

If you have a suspended task, treat this as a wait problem. Use the techniques of Chapter 6, "Dealing with waits" on page 31 to investigate it further.

If you have a running task, it is likely to be looping. Refer to Chapter 7, "Dealing with loops" on page 39 to find out what to do next.

What to do if the task is not in the system

If you have obtained no output and the CECI INQ TASK command shows that the task is not in the system, one of two things could have happened:

- Your transaction never started.
- Your transaction ran, but produced no output.

Note: If you're not getting output on a printer, the reason could be simply that you are not setting on the START PRINTER bit in the write control character. You need to set this bit to get printed output if you have specified the STRFIELD option on a CONVERSE or SEND command, which means that the data area specified in the FROM option contains structured fields. Your application must set up the contents of the structured fields.

Your task might have been initiated by a direct request from a terminal, or by an automatic transaction initiation (ATI). Most of the techniques apply to both sorts of task, but there are some extra things to investigate for ATI tasks. Carry out the tests that apply to all tasks first, then go on to the tests for ATI tasks if you need to.

Did the task run?

There are many different techniques for finding out whether a transaction started, or whether it ran but produced no output. Use the ones that are most convenient at your installation.

Using CICS/400 system trace entry points

CICS system tracing is probably the most powerful technique for finding out whether a transaction ever started. You might need to direct the trace output to the auxiliary trace destination, depending on how certain you can be about the time the task is expected to start. Even a large internal trace table might wrap and overlay the data you want to see if you are not sure about when the task should start.

Now turn tracing on, and attempt to start your task. When you are sure that the time has passed when the output should have appeared, stop tracing, and format the trace entries. Use the PRTCICSTRC CL command to format the trace entries.

If your transaction ran, you should see the following types of trace entries for your task and the programs associated with it:

- Application shell (AEGCSxxx) entries showing the setup for the task
- Program control (AEGPCPGM) link to your application program

- EXEC interface (AEGEIEIP) entries for each EXEC CICS command executed within your application
- CICS functional component trace entries (for example, AEGFCxxx – file control, AEGKCxxx – task control, AEGICxxx – interval control) for the CICS service functions being executed to process the CICS request
- CICS syncpoint (AEGSPxxx) trace entries at task termination
- Application shell (AEGCSxxx) entries showing task cleanup functions

Depending on when CICS trace was activated during your task, all, some, or none of these trace entries may appear in the formatted trace output.

Using EDF

If the transaction being tested requires a terminal, you can use EDF. You need two terminals for input, in addition to the one the transaction requires ("tttt"). Use one of these additional terminals to put the transaction terminal under control of EDF, with:

```
CEDF tttt
```

At the remaining terminal, enter whatever transaction or sequence of transactions causes the one under test to be initiated. Wait long enough for it to start. If no output appears at the second terminal, the transaction has not started. If you haven't yet done so, consider using trace to get more information about the failure.

Using CEBR

You can use the CEBR transaction to investigate your transaction if the transaction reads or writes to a transient data queue, or writes to a temporary storage queue. A change in such a queue is strong evidence that the transaction ran, if the environment is sufficiently controlled that nothing else could produce the same effect. You need to be sure that no other transaction that might be executed while you are doing your testing does the same thing.

The absence of such a change does not mean that the transaction did not run: it may have run incorrectly, so that the expected change was not made.

Using the command-level interpreter CECI

If your transaction writes to a file, you can use the command-level interpreter CECI before and after the transaction to look for evidence of the execution of your transaction. A change in the file means that the transaction ran. If no change occurred, that does not necessarily mean that the transaction failed to run–it could have worked incorrectly, so that the changes you were expecting were not made.

Disabling the transaction

If your transaction requires a terminal, you can do the following. Use the CEMT transaction to disable the transaction under test, then do whatever causes the transaction to be initiated. You should get the following message at the terminal where it is due to run:

```
The CICS transaction tranid could not be started due to reason 84
```

You should test the EIBRESP field for this return code. See the *CICS for iSeries Application Programming Guide* for information about the EIBRESP field.

If you don't get this message, it is likely that your transaction did not start because of a problem with that terminal.

Investigating tasks initiated by ATI

In addition to the general techniques for all tasks described above, there are some additional ones for tasks that were started by ATI.

Tasks to be started by ATI can be invoked in either of the following ways:

- By issuing EXEC CICS START commands, even if no interval is specified
- By writing to transient data queues with nonzero trigger levels

There are many reasons why automatically initiated tasks could fail to start. Even when the CICS system is operating normally, an ATI task might fail to start if a resource that it requires is not available. The resource is usually a terminal, although it could be a queue.

If you know the CICS terminal identifier (TERMID value), perform a CEMT INQ TERMINAL command to see whether you tried to start a task against a NOATI terminal, or whether the terminal is currently being used by another task. In either of these cases, your START request will not begin processing.

If the START request is for a nonterminal facility, CICS/400 submits interval control batch jobs identified by a job name of AEGxxxxICB where xxxx is replaced with the CICS control region identifier. Examine the job queues using the WRKSBMJOB CL command to see whether there are IC batch jobs awaiting execution.

If you find an AID that relates to your task, your task is scheduled to start but cannot do so because the terminal is unavailable. Look in field AIDTRMID to find the symbolic ID of the terminal, and then investigate why the terminal is not available. One possibility is that the terminal is not in ATI status, because ATI(YES) has not been specified for it in the TCT definition.

Your transaction produced wrong output

If your transaction produced no output at all, read "Your transaction produced no output at all" on page 53. For other types of wrong terminal output, read the following sections.

The origins of corrupted data

You get incorrect output to a terminal if data that is the object of the transaction becomes corrupted at some stage.

Figure 5 on page 57 illustrates how data flows between various CICS resources when a transaction is executed, and shows the points at which the data might become invalid.

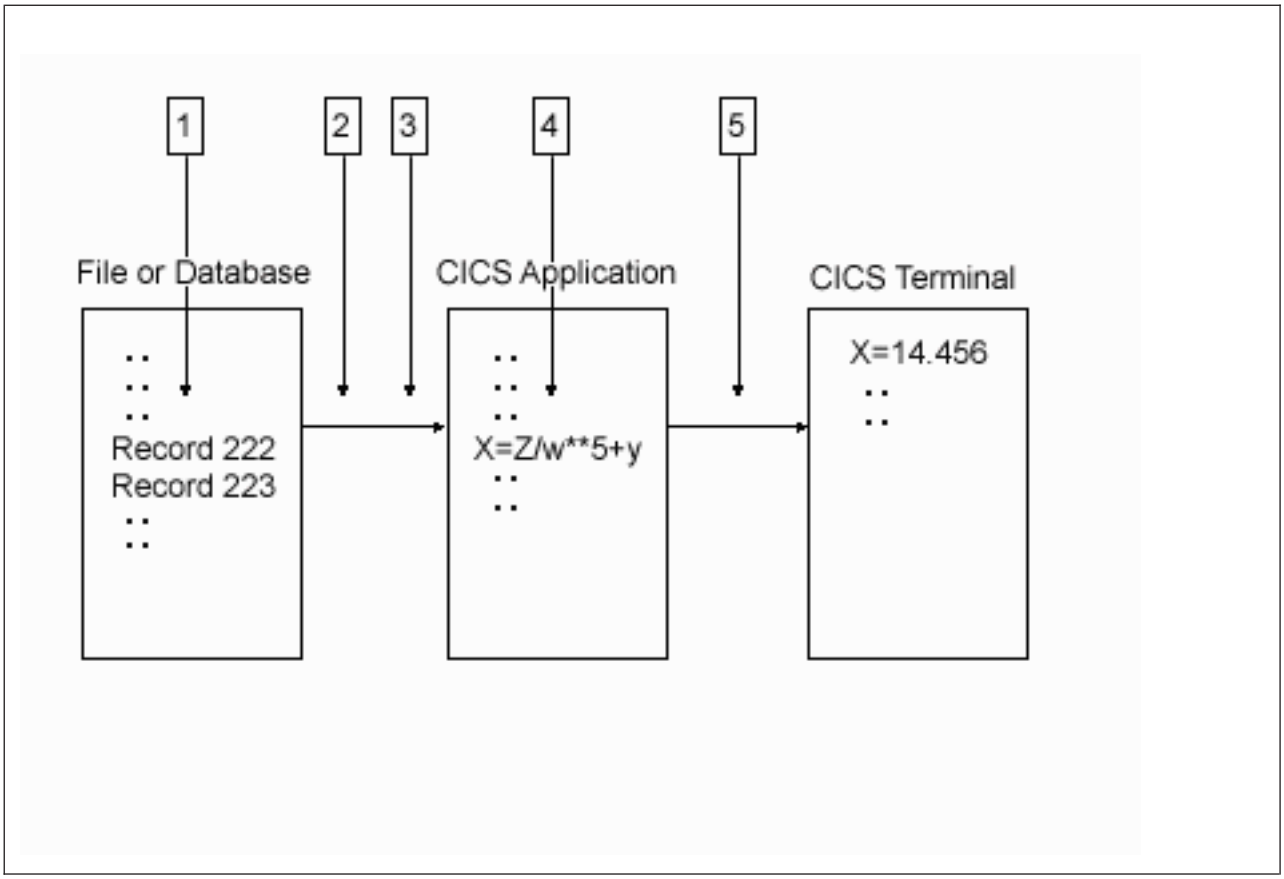


Figure 5. Where data might be corrupted in a transaction

The data might be corrupted at any of points 1 through 5, as it flows from the file to the terminal:

1. Data records might be incorrect, or they could be missing from the file.
2. Data from the file might be mapped into the program incorrectly.
3. Data input at the terminal might be mapped into the program incorrectly.
4. Incorrect programming logic might corrupt the data.
5. The data might be mapped incorrectly to the terminal.

Each of these possibilities is dealt with in turn in the following sections.

Were records in the file incorrect or missing?

You can check the contents of a file or database by using the command-level interpreter CECI, or by using a utility program, or by using a CL command to list off the records in question.

If you find invalid data in the file, the error is likely to have been caused by the program that last updated the records containing that data. If the records you expected to see are missing, make sure that your application can deal with a 'record not found' condition.

If the data in the file is valid, it must have been corrupted later on in the processing.

Was the data mapped correctly into the program?

When a program reads data from a file or a database, the data is put into a field described by a symbolic data declaration in the program.

Is the data contained in the record that is read compatible with the data declaration in the program?

Check each field in the data structure receiving the record, making sure in particular that the type of data in the record is the same as that in the declaration, and that the field receiving the record is the right length.

If the program receives input data from the terminal, make sure that the relevant data declarations are correct as well.

If there seems to be no error in the way in which the data is mapped from the file or terminal to the program storage areas, the next thing to check is the program logic.

Is the data being corrupted by incorrect programming logic?

To find out if data is being corrupted by incorrect programming logic in the application, consider the flow of data through the transaction.

You can determine the flow of data through your transaction by "desk checking", or by using the interactive tools and tracing techniques supplied by CICS.

Desk checking your source code is sometimes best done with the help of another programmer who is not familiar with the program. It is often possible for such a person to see weaknesses in the code that you have overlooked.

Interactive tools allow you to look at the ways in which the data values being manipulated by your program change as the transaction proceeds.

- The ILE source debugger is a powerful and easy-to-use tool that can help you locate programming compile-time errors in C programs. For details, see the *WebSphere Development Studio: ILE C/C++ Programmer's Guide*.
- The CEDF transaction is a CICS-supplied, interactive tool for checking your programming logic. You can use it to follow the internal flow from one CICS command-level statement to another. If necessary, you can add CICS statements such as ASKTIME at critical points in your program, to see if certain paths are taken, and to check program storage values.
- The CEDF transaction can be used with native OS/400 debugging (STRDBG) to provide a more thorough debugging session. Refer to the *WebSphere Development Studio: ILE C/C++ Programmer's Guide* for further information.
- The command-level interpreter CECI allows you to simulate CICS command statements. Try to make your test environment match as closely as possible the environment in which the error occurred. If you don't, you might find that your program works with the command-level interpreter CECI, but not otherwise.
- The CEBR transaction enables you to look at temporary storage and transient data queues, and to put data into them. This can be useful when many different programs use the queues to pass data.

  Note: When you use the CEBR transaction to look at a transient data queue, the records you retrieve are removed from the queue before they are displayed to you. This could alter the flow of control in the program you

are testing. You can, however, use the CEBR transaction to copy transient data queues to and from temporary storage, as a way of preserving the queues if you need to.

The above tools are described in more detail in the *CICS for iSeries Application Programming Guide*.

User tracing allows you to trace the flow of control and data through your program, and to record data values at specific points in the execution of the transaction. You could, for example, look at the values of counters, flags, and key variables during the execution of your program. You can include up to 4000 bytes of data on any trace entry, and so this can be a useful technique for finding where data values are being corrupted.

Is the data being mapped incorrectly to the terminal?

Incorrect data mapping to a terminal can have both application-related and system-related causes. If you are using BMS mapping, check the following items:

- Examine the symbolic map very carefully to make sure that it agrees with the map in the load module. Check the date and time stamps, and the size of the map.
- Make sure that the attributes of the fields are what they should be. For example:
  - An attribute of DARK on a field can prevent the data in the field from being displayed on the screen.
  - Not turning on the modified-data tag (MDT) in a field might prevent that field from being transmitted when the screen is read in.

    Note: The MDT is turned on automatically if the operator types data in the field. If, however, the operator does not type data there, the application must turn the tag on explicitly if the field is to be read in.
- If your program changes a field attribute byte or a write control character, look at each bit and check that its value is correct by looking in the appropriate reference manual for the terminal.

Chapter 10. Dealing with storage violations

This chapter covers the following:

Storage violations can be divided into two classes: those detected and reported by CICS, and those not detected by CICS. They require different problem determination techniques.

CICS-detected violations are identified by the following message being sent to the SYSOPR queue:

```
AEG0504 Storage control has detected a serious error
```

Storage violations not detected by CICS are less easy to identify. They can cause almost any sort of symptom. Typically, you may have got a program check with a condition code indicating 'a machine check' or 'data exception', because the program or its data has been overlaid. Otherwise, you might have obtained a message from the dump formatting program saying that it had found a corrupted data area. Whatever the evidence for the storage violation, if it has not been detected by CICS, turn to "Storage violations that affect innocent transactions" on page 63.

CICS has detected a storage violation

CICS can detect storage violations when the leading storage check zone or the trailing storage check zone of a user-task storage element has become corrupted. CICS detects storage violations involving user-task storage when it receives a command to free that element of storage. It also checks the chains when it frees all the storage belonging to a task when the task terminates. The storage violation is detected only when the storage check zones are validated. This is illustrated in Figure 6 on page 62, which shows the sequence of events when CICS detects a violation of user-task storage.

The storage check zone is overlaid in user storage. Usually this does not matter, because the transaction whose storage is violated is normally responsible for the violation. However, if the violation occurs in shared storage, it is likely that the transaction causing the violation is longer in the system.

What happens when CICS detects a storage violation

When CICS detects a storage violation, it makes an error trace entry, issues a message, and takes a CICS transaction dump. In addition, the transaction detecting the violation may fail with a storage control program (ASCP) abend.

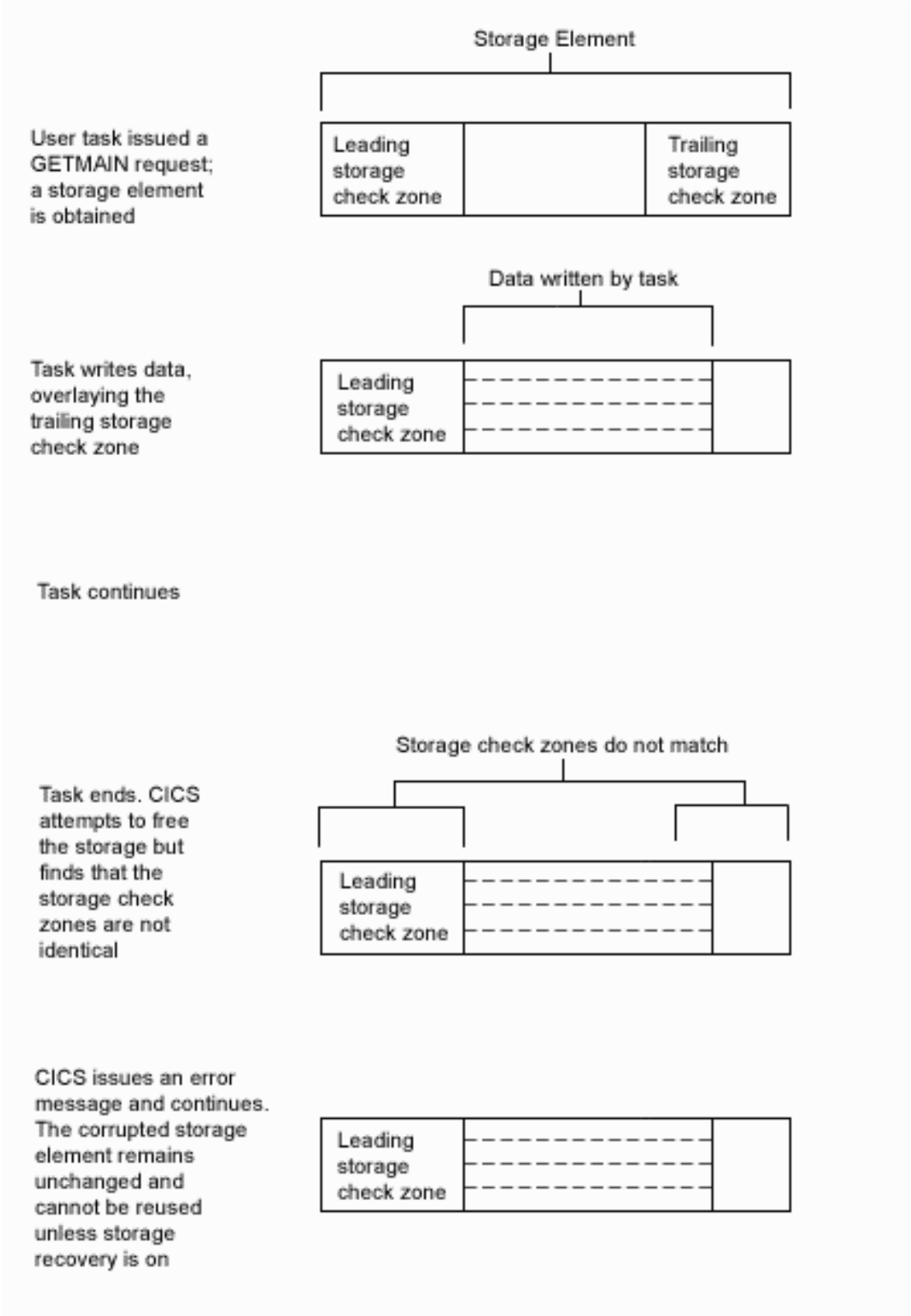


Figure 6. How user-storage violations are committed and detected

What the transaction abend message can tell you

If you get a transaction abend message, it is very likely that CICS detected the storage violation when it was attempting to satisfy a FREEMAIN request for user storage. Make a note of the information the message contains, including:

- The transaction abend code
- The identity of the transaction whose storage has been violated
- The identity of the program running at the time the violation was detected
- The identity of the terminal at which the task was started

Because CICS does not detect the overlay at the time it occurs, the program identified in the abend message probably is not the one in error. However, it is likely that it issued the FREEMAIN request on which the error was detected. One of the other programs in the abended transaction might have violated the storage in the first place.

What CICS trace can tell you

Look for trace entries recorded by the storage control programs for FREEMAIN (AEGSCFMA) and GETMAIN (AEGSCGMA). An error trace entry written by AEGSCFMA indicates a problem in performing a FREEMAIN request. This error trace entry identifies the task number where the error occurred. Format the auxiliary trace again, selecting only the identified task. The selective task trace provides a more concise view of what the specific task was doing to cause the storage violation to occur.

Storage violations that affect innocent transactions

Storage violations that affect innocent transactions—that is, transactions that do not cause the violation—usually go undetected by CICS. However, occasionally CICS detects the storage check zone of a shared-storage element that has been overlaid by a task that does not own it.

If they are reproducible, storage violations of this type typically occur at specific offsets within structures. For example, the start of an overlay might always be at offset 30 from the start of a field.

The most likely cause of such a violation is a transaction that is writing data to a part of the shared storage that it does not own, or possibly freeing such an area. The transaction might previously have obtained the area and then freed it before writing the data. Or addressability might not have been correctly maintained by the application.

Storage violations affecting innocent transactions are, in general, more difficult to resolve than those that are detected by CICS. Often, you become aware of them long after they occur; then you need a long history of system activity to find out what caused them.

A strategy for storage violations affecting innocent transactions

If the storage violation has been caused by a program writing to an area that it does not own, you probably have no idea at the outset which program is at fault. Look carefully at the content of the overlay before you do any other investigation, because this could help you to identify the transaction, program, or routine that

caused the error. If it does not provide the clue you need, your strategy should be to use CICS tracing to collect a history of all the activities that referred to the affected area.

The trace table must go back as far as the start of the program causing the overlay. This could mean that a very large trace table is needed. Internal tracing is not suitable because it wraps when it is full, and thus overwrites important trace entries.

Auxiliary trace objects are suitable destinations for recording long periods of system activity, because they don't wrap when they are full.

If you have no idea which transaction is causing the overlay, you need to trace the activities of every transaction. This impacts performance, because of the processing overhead.

A procedure for resolving storage violations affecting innocent transactions

Select auxiliary trace as the trace destination. When you get the symptoms that tell you that the storage violation has occurred, stop the trace and format the auxiliary trace using the PRTCICSTRC CL command.

The next job is to locate all the entries in the trace table that address the overlaid area. Operations involving GETMAIN and FREEMAIN requests in particular are likely pointers to the cause of the error. Look for GETMAIN (AEGSCGMA) and FREEMAIN (AEGSCFMA) trace entries.

When you have found a likely trace entry, possibly showing a GETMAIN or FREEMAIN request addressing the area you have identified the suspected task number. Rather than locating this manually, it is probably better to reformat the auxiliary trace selectively to show just the trace entries corresponding to the task number.

Having found the identity of the transaction, look at all the programs belonging to the transaction. It is likely that one of these caused the overlay, and you need to consider the logic of each to see if it could have caused the error. This is a long job, but it is one of the few ways of resolving a storage violation affecting an innocent transaction.

What to do if you still can't find the cause of the overlay

If you are unable to identify the cause of the storage violation after carrying out the procedures of the preceding section, contact your IBM Support Center. They may suggest a method for detecting the storage violation.

Programming errors that can cause storage violations

There are a number of commonly occurring programming errors that can cause storage violations.

- Failing to obtain sufficient storage. This is often caused by failure to recompile all the programs for a transaction after a common storage area has been redefined with a changed length.
- Runaway subscript. Make sure that your tables can grow only to a finite size.

- Writing data to an area after it has been freed. When a task frees an area that it has been addressing, it can no longer write data to the area without the risk of overwriting some other data that might subsequently be there.

Part 3. Using traces and dumps in problem determination

Chapter 11. Using dumps in problem determination

This chapter covers the following:

You have the choice of two different types of CICS dumps to help you with problem determination. They are the **transaction dump**, of transaction-related storage areas, and the **CICS system dump**, of the entire CICS control region.

The type of dump to use for problem determination depends on the nature of the problem. In practice, the system dump is more comprehensive because it contains more information than the transaction dump. You can be reasonably confident that the system dump has captured all the evidence you need to solve your problem. The system dump is particularly useful if you find you need the assistance of your IBM Support Center. However, the transaction dump along with any error messages is a good place to begin your investigations.

The amount of CICS system dump data that you could get is potentially very large, but that need not be a problem. You can leave the data on the system dump object, or keep a copy of it, and format it selectively as you require.

You can control the dump actions taken by CICS, and also the information the dump output contains.

Dump output is incorrect

Read this section if you did not get the dump output you were expecting.

Indications that the dump output could be in error are as follows:

- The dump did not seem to relate to the CICS control region you were interested in
- You did not get a dump when an abend occurred

The sections that follow give guidance about resolving both of these problems in turn.

Dump did not relate to CICS control region

If you have experienced this problem, it is likely that you have dumped the wrong CICS control region. This problem should not occur if you are running a single control region.

If you invoked the CEMT PERFORM SNAP command from within a control region, a spool file called DPFHDUMP is created with spool file user data of CICSCRDUMP. This dump represents the output of a DMPJOB CL command, dumping all the control region. If this dump does not represent the control region you expected, then the CEMT PERFORM SNAP command was executed from a different control region. Log on to the correct control region using the STRCICSUSR CL command and request the snap dump again.

You did not get a dump when an abend occurred

Read this section if you have experienced either of the following problems:

- A transaction abended, but you did not get a transaction dump
- A system abend occurred, but you did not get a system dump

There are, in general, two reasons why dumps might not be taken:

- Dumping was suppressed because of the way the dumping requirements for the CICS control region were defined. The valid ways that dumping can be suppressed are described in detail in the sections that follow.
- If no dump was created when a control region abend occurred, an error message indicating the problem appears in the control region job log and in the QSYSOPR message queue.

How dumping can be suppressed

If you did not get a dump when an abend occurred, and there was no system error, the dumping you required must somehow have been suppressed. There are two levels at which dumping can be suppressed:

- System dumps can be globally suppressed
- Transaction dumps can be suppressed for individual transactions

You need to find out which of these types of dump suppression apply to your system before you decide what remedial action to take.

Global suppression of system dumping

System dumping can be suppressed globally in two ways:

- By coding a value of *NO for the DUMP system initialization parameter
- By using the system programming command SET SYSTEM DUMPING with a CVDA value of NOSYSDUMP

You can inquire whether system dumping has been suppressed globally by using the INQUIRE SYSTEM DUMPING system programming command. If necessary, you can cancel the global suppression of system dumping using the SET SYSTEM DUMPING command with a CVDA value of SYSDUMP.

Suppression of dumping for individual transactions

Transaction dumps taken when a transaction abends can be suppressed for individual transactions by using the SET TRANSACTION DUMPING system programming command, or by using the DUMP attribute on the ADDCICSPCT CL definition of the transaction.

You can use the INQUIRE TRANSACTION DUMPING command to see whether dumping has been suppressed for a transaction, and then use the corresponding SET command to cancel the suppression if necessary.

Controlling dump action

There are two aspects to controlling the dump action:

- Setting up the dumping environment, so that appropriate dump action is taken when circumstances arise that might cause a dump to be taken.
- Causing a dump to be taken. Users and CICS can issue requests for dumps to be taken.

Setting up the dumping environment

There are several levels at which the dumping environment can be set up:

- System dumps can be globally suppressed or enabled at system initialization by using the DUMP system initialization parameter.
- System dumps can be globally suppressed or enabled dynamically through the SET SYSTEM DUMPING command.
- Transaction dumps can be suppressed or enabled dynamically for individual transactions. This is done by using the SET TRANSACTION DUMPING command, or by using the DUMP attribute of the PCT definition for the transaction.

Events that can cause dumps to be taken

The following are the events that can cause dumps to be taken, if the dumping is allowed under the circumstances:

- Explicit requests for dumps from users
- CICS transaction abends
- CICS system abends

On most occasions when dumps are requested, CICS refers to a dump code that is specified either implicitly or explicitly to determine what action should be taken.

The ways that you can request dumps

You can issue an explicit request for a dump by using the CEMT transaction, or by using an EXEC CICS command.

- The CEMT PERFORM SNAP command enables you to get a CICS system dump, if system dumping has not been globally suppressed.
- You can use the PERFORM SHUTDOWN DUMP command to shut down the control region and to get a CICS system dump, if system dumping is not globally suppressed.
- You can use the DUMP TRANSACTION command to get a transaction dump. You get a transaction dump even if dumping has been suppressed for the transaction that you identify on the command.

  You must specify a transaction dump code when you use this command and it must be a maximum of 4 characters in length. It could, for example, be TD01.

You might use these methods of taking dumps if, for example, you have a task in a wait state, or you suspect that a task is looping. In the event of a transaction abend, or a system abend, a dump is taken automatically, provided that DUMP(*YES) has been specified in the SIT.

In order for you to take a dump, either by issuing an EXEC CICS PERFORM DUMP command, or when your transaction ends, you need the appropriate authority. You can obtain this in the following ways:

- Preferably, use the GRTOBJAUT command to grant object authority over the DMPJOB command in QSYS:

  ```
  GRTOBJAUT OBJ(QSYS/DMPJOB) OBJTYPE(*CMD) USER(*PUBLIC) AUT(*USE)
  ```

- Include *ALLOBJ in your user profile. This option is not recommended, because it would give you authority for every object on the system.

If you do not have the correct authority for either of these actions, see your system administrator.

The occasions on which CICS requests a dump

In general, CICS requests a dump when a transaction or CICS system abend occurs. The dumping environment determines whether or not a dump is actually taken.

The following are the occasions when CICS requests a dump:

- CICS requests a transaction dump, and perhaps a system dump, after a transaction abend occurs. There are two cases:
  - You can include the ABEND command in one of your applications, causing it to abend when some condition occurs. You must specify a four-character transaction abend code on this command, and this is used as the transaction dump code. It could, for example, be MYAB.
  - CICS might cause a transaction to abend. In this case, the four-character CICS transaction abend code is used as the transaction dump code. See Appendix B, "Abend codes" on page 93, and the *CICS for iSeries Application Programming Guide*, for a list of CICS/400 abend codes and conditions.
- A CICS system dump could be taken following a CICS system abend. In this case, the user data of the DPFHDMP spool file contains "CICSCRDUMP".

Specifying the areas you want written to a transaction dump

When you use the DUMP TRANSACTION command to get a transaction dump, you receive either a COBOL/400 formatted dump or user-shell oriented DMPJOB output, depending on the language convention of the active PPT entry. You always get a complete transaction dump whenever a transaction abend occurs, if the transaction requires a transaction dump to be taken.

Looking at dumps

CICS/400 creates a spool file called DPFHDMP in the output queue of the job in which the dump was requested. A WRKSPLF (Work with Spooled Files) CL command can be used to identify CICS dumps. In addition, the user data area of the spooled file indicates either the CICS abend code for the dump, or a literal "CICSCRDUMP" indicating a CICS control region dump.

CICS/400 transaction dump

A CICS/400 transaction dump consists of a COBOL/400 formatted dump for the COBOL program active at the time the dump was requested. The COBOL/400 formatted dump indicates where the error occurred in the COBOL program. This dump also lists in alphabetic order all the COBOL data names within the COBOL program.

CICS/400 system dump

A CICS system dump consists of the maximum output available from a DMPJOB CL command, executed in the control region's job. These dumps may be useful for the IBM Support Center to resolve problems. Error messages issued on behalf of CICS/400 are much better than CICS system dumps as an aid to resolving problems.

Chapter 12. Using traces in problem determination

Several types of trace are available to record different aspects of CICS activities. You can control the destinations of CICS trace entries, and the activation or deactivation of the various trace destinations. In addition, you can control whether trace entries requested from application programs (user tracing) are recorded in active CICS trace destinations. This chapter covers the following:

Trace spool files are always produced at the current release level. The current release is printed in the top left-hand corner of the report, on the first heading line. See Figure 7 on page 80.

CICS/400 provides three tracing facilities:

- Internal tracing
- Auxiliary tracing
- User tracing

Trace output is incorrect

If you have been unable to get the trace output you need, you can find guidance about solving this problem in this section. You can be selective about the way CICS does tracing; options must be considered carefully to make sure you get the tracing you need.

There are two main types of problem:

- Your tracing might have gone to the wrong destination. This is dealt with in "Tracing has gone to the wrong destination".
- You might have captured the wrong data. This is dealt with in "You have captured the wrong trace data" on page 74.

Tracing has gone to the wrong destination

All CICS trace entries go to the internal trace table and to any other trace destination that is currently active.

Internal tracing goes to the internal trace table in main storage. Trace entries are written to any trace destination only if internal trace is active. No trace entries are written to any trace destination if internal trace is off.

Auxiliary tracing goes to one of two user space (*USRSPC) objects, if the auxiliary tracing status is STARTED. What happens when the auxiliary trace *USRSPC becomes full is determined by the auxiliary switch status. You need to make sure

that the switch status is correct for your system, or you might lose the trace entries you want, either because the object is full or because they are overwritten.

Trace entries written by EXEC CICS ENTER commands are referred to as **user trace entries**. User trace entries appear in the active trace destinations only if the user trace status is active.

You have captured the wrong trace data

There are several ways in which you could capture the wrong trace data. The following are some symptoms that suggest this type of problem:

- The user trace entries, output from your application, do not appear in the trace data.
- The trace points selected do not adequately trace some components.
- The formatted trace print indicates that no trace entries have been selected for print.
- The entries you want are missing entirely from the formatted trace output.

  If you have this sort of problem, refer to "The entries you want are missing from the trace table" for guidance about finding the cause.

It is worth remembering that the more precisely you can define the trace data you need for any sort of problem determination, the more quickly you are likely to find the cause of your problem.

The entries you want are missing from the trace table

Read this section if one or more entries you were expecting were missing entirely from the trace table.

The following cases are considered:

- The trace data you wanted did not appear at the expected time.
- The earliest trace entry in the table was time-stamped after the activity that you were interested in took place.
- You could not find the exception trace entries you selected.

If the trace entry did not appear at the expected time, consider whether you entered the correct transaction and terminal identifier. This is most likely to happen if you attempted to format the auxiliary trace object using the PRTCICSTRC command selectively by transaction or terminal.

When you select trace entries for formatting by specifying TRANSID or TERMID values in the PRTCICSTRC command, CICS trace function searches for any trace entries that contain the specified TRANSID or TERMID values. The trace function then formats any trace entries that meet the selection criteria.

For guidance information about trace formatting using the PRTCICSTRC CL command, see Chapter 13, "PRTCICSTRC (Print CICS Trace) command" on page 85.

If the options were correct and tracing was running at the right time, but the trace entries you wanted did not appear, it is likely that the task you were interested in did not run. Examine the trace carefully in the control region in which you expected the task to appear, and attempt to find why it was not invoked.

If the earliest trace entry was later than the event that interested you, and tracing was running at the right time, it is likely that the trace table wrapped around and earlier entries were overwritten.

Internal trace always wraps when it is full. Try using a bigger trace table, or direct the trace entries to the auxiliary trace.

Note: Changing the size of the internal trace table during a run causes the data that was already there to be destroyed. In such a case, the earliest data would have been recorded after the time when you redefined the table size.

Auxiliary trace switches from one object to the next when it is full, if the autoswitch status is NEXT or ALL.

If the autoswitch status is NEXT, the two *USRSPCs can fill up but earlier data cannot be overwritten. Your missing data might be in the initial object, or the events you were interested in might have occurred after the objects were full. In the second case, you can try increasing the size of the auxiliary trace objects.

If the autoswitch status is ALL, you might have overwritten the data you wanted. The initial object is reused when the second extent is full. Try increasing the size of the auxiliary trace objects.

If the formatted trace print indicates that no trace entries have been selected for print, check the following:

- Did you enter the wrong object name in the PRTCICSTRC command?
- Did you enter an object name of the auxiliary trace object that was inactive at the time the task was executed?
- Did you enter an object name that is for an auxiliary trace object that is not being used by the control region in which your transaction was executed?

Internal tracing

Internal tracing is the recording of trace entries to a **control region internal trace table**. This trace table is a special storage class (TRACETABLE). Trace modules manage this storage class.

The TRACETABLE storage class is a *USRSPC object created in the QTEMP library of the control region. This object, because it resides in QTEMP, is deleted when the control region is terminated. **First Failure Data Capture** (FFDC) routines attempt to dump this space object in the event of a control region system abend.

Internal tracing can be made active at system initialization through the INTTRCCTL parameters in the ADDCICSSIT and CHGCICSSIT commands. In addition, internal tracing can be activated or deactivated using the CEMT SET INTTRACE command, or from within CICS application programs using the EXEC CICS SET TRACEDEST command. Refer to the *CICS for iSeries Administration and Operations Guide* for information about the CEMT SET INTTRACE command and the *CICS for iSeries Application Programming Guide* for the EXEC CICS SET TRACEDEST command.

It is important to note that all of CICS/400 tracing is controlled by the internal trace setting. If internal trace is not active, no CICS trace entries are recorded to any CICS trace destinations.

Auxiliary tracing

Auxiliary tracing (auxtrace) is a recording of trace entries to auxiliary trace *USRSPC objects. These objects are defined to the control region through the system initialization table (SIT) option AUXTRCCTL (in ADDCICSSIT, CHGCICSSIT commands).

Two *USRSPC objects may be allocated per control region. The trace user space objects named in the SIT are created at control region startup if they do not already exist. Dynamic switching capabilities of these auxiliary trace areas are provided by CEMT INQUIRE|SET AUXTRACE and EXEC CICS INQUIRE|SET TRACEDEST commands.

Auxiliary tracing may be activated at control region startup by using AUXTRCCTL(*YES), or may be activated or deactivated in a manner similar to activating internal trace. **However, no trace entries are recorded to the auxiliary trace objects unless CICS internal trace is also active.**

User tracing

User tracing allows the recording of user-defined trace entries to CICS/400 trace (internal and auxtrace) areas. User trace entries are recorded only if user and internal tracing are active within a control region.

User trace entries are recorded as a result of ENTER commands within an application program.

As with auxiliary trace, the USRTRCCTL options of the ADDCICSSIT and CHGCICSSIT commands control user tracing at control region startup. After control region startup, CEMT INQUIRE|SET INTTRACE|AUXTRACE and EXEC CICS INQUIRE|SET TRACEDEST commands can be used to inquire about, activate, or deactivate CICS user tracing.

CICS tracing records

CICS tracing can be used to trace the flow of execution through CICS code, and through your application code as well. For information about how to use trace calls from within your application, see the *CICS for iSeries Application Programming Guide*. You can see what functions are being performed, what parameters are being passed, and the values of important data fields at the time trace calls are made.

Trace entry types

There are six types of trace entry. Each trace entry type can be easily identified within an AUXTRACE *USRSPC object or within the TRACETABLE internal trace object of the control region.

Entry type	Use
ENTRY	Written by each CICS module at entry to the routine.
EXIT	Written by each CICS module just before exiting.
EVENT	Written by certain CICS modules to identify special events within that module. Examples of these could include: • Event recorded just before calling a COBOL application program

- Event recorded before calling an OS/400 routine to perform a service on behalf of CICS

Event trace entries track the flow of control within a CICS module. Event trace numbers are unique within a specific CICS module.

ERROR Written by a CICS/400 module when an error has been detected from which the module cannot recover. Normally, an error message is written to the QSYSOPR message queue for these types of error. In addition, CICS first failure data capture may be performed as a result of errors detected.

USER Written as the result of a valid ENTER command being successfully executed.

LOCK Written to indicate temporary locking of an internal CICS resource.

Selecting trace destinations and related options

The trace destinations you select and the options you define for the destinations depend on your specific problem determination requirements. You can select any combination of CICS internal tracing or CICS auxiliary tracing. Your decision must be based on the characteristics of the various types of CICS tracing, which are described in "CICS internal trace" and "CICS auxiliary trace" on page 78. You must decide how much trace data you need to capture, and whether you want to integrate CICS tracing with tracing done by other programs.

You can control the status and certain other attributes of the various types of CICS tracing as follows:

- During system initialization, using system initialization parameters.
- Dynamically, using the CEMT SET TRACE START³STOP transaction or the EXEC CICS SET TRACEDEST command. For example, you can use the EXEC CICS SET TRACEDEST TABLESIZE command to increase the size of the internal trace table.

CICS internal trace

You can control the settings of internal trace at control region startup using the INTTRCCTL parameter of the ADDCICSSIT or CHGCICSSIT commands. You can select, at startup, a status of ACTIVE(*YES or *NO) for internal trace. If internal trace is active at startup, any trace calls that are made cause trace entries to be directed to the internal trace table.

The internal trace table has a minimum of 125 entries, and a maximum of 10 000 entries.

Note: You can change the size of the table dynamically, while CICS is running, but if you do so, you lose all of the trace data that was present in the table at the time of the change.

As a default, the internal trace table wraps when it is full. When the end of the table is reached, the next entry to be directed to the internal trace entry goes to the start, and overlays the trace entry that was formerly there. In practice, the internal trace table cannot be very big, so it is most useful for background tracing or when you don't need to capture an extensive set of trace entries. If you need to trace CICS system activity over a long period, or if you need many entries over a short period, auxiliary tracing is likely to be more appropriate.

CICS auxiliary trace

CICS auxiliary trace entries are directed to one or the other of two auxiliary trace objects. These are CICS owned objects, and they must be created before CICS is started. They cannot be redefined dynamically.

You can use the Active at Startup element of the AUXTRCCTL parameter of ADDCICSSIT or CHGCICSSIT to turn CICS auxiliary trace on or off at startup.

You can select a status of AUXSTART, AUXSTOP, or AUXPAUSE for CICS auxiliary trace dynamically using the CEMT INQUIRE|SET AUXTRACE transaction. These statuses reflect both the value of the auxiliary trace flag, and the status of the current auxiliary trace object, in the way shown in Table 2.

Table 2. The meanings of auxiliary trace status values

Auxiliary trace status	Auxiliary trace flag	Auxiliary trace object
STARTED	ON	ACTIVE
STOPPED	OFF	INACTIVE
PAUSED	OFF	ACTIVE

When you first select AUXSTART for CICS auxiliary trace, any trace entries are directed to the initial auxiliary trace object. If you initialize CICS with auxiliary trace ACTIVE, the initial auxiliary *USRSPC object specified is used.

What happens when the initial object is full depends on the auxiliary switch status. This can have a value of *YES or *NO, and you can define it either in the system initialization table, using the Automatic Switching element of the AUXTRCCTL system initialization parameter (see the *CICS for iSeries Administration and Operations Guide*), or dynamically using the CEMT transaction.

***NO** means that when the initial object is full, no more auxiliary tracing is done.

***YES** means that auxiliary trace data is written alternately to each object, a switch being made from one to the other every time the current one becomes full. This means that trace entries already present in the trace objects start getting overwritten when both objects become full for the first time.

You can determine which is the current trace object by using the command DSPLOG MSGID(AEG1403) to display messages issued when auxiliary trace switching occurs.

The advantage of using auxiliary trace is that you can collect large amounts of trace data, if you initially define large enough trace objects. For example, you might want to do this to trace system activity over a long period of time, perhaps to solve an unpredictable storage violation problem.

A time stamp is included in the header line of every page of abbreviated auxiliary trace output to help match external events with a particular area of the trace, and thus help you to find the trace entries that are of interest.

Setting the tracing status

You can set the system tracing status by coding the appropriate system initialization parameters, and you can also set it dynamically using the CEMT transaction.

Setting the tracing status at system initialization

The parameters that you can use to set up tracing status at system initialization are: INTTRCCTL for internal trace, AUXTRCCTL for auxiliary trace, and USRTRC for user trace, see *CICS for iSeries Administration and Operations Guide*.

Setting the tracing status using the CEMT transaction

You can use the CEMT SET INTTRACE commands to set the tracing status.

The internal trace table size is 125 entries, which is the minimum size it can be. If internal trace were STARTED, the trace table would wrap when it became full.

If the current auxiliary object is B, this means that trace entries would be written to object B if auxiliary tracing were started. However, if its status is shown to be PAUSED, no tracing is done to that destination. If the auxiliary switch status is ALL, a switch would be made to the other auxiliary trace object whenever one became full.

If the user trace flag is OFF, no user trace entries can be made from applications. You must set the user trace flag on before any user trace requests in your programs can be serviced. If it were off, any trace requests in your programs would be ignored.

Formatting CICS trace entries

The PRTCICSTRC command is provided for formatting trace entries from any CICS auxiliary trace *USRSPC object. This command provides the capability to format and print CICS trace entries selectively. See Chapter 13, "PRTCICSTRC (Print CICS Trace) command" on page 85 for more information about the use of the PRTCICSTRC command.

You can specify *BASIC or *FULL trace formatting, to give you a formatted trace print of either one or several lines per trace entry. Remember that each trace entry may be up to 4000 bytes of trace data. CICS internally may write large trace entries. You may occasionally want to look at the extended information to understand in more detail the information given in the corresponding *BASIC entries.

You can control the formatting and you can select the trace entries on the basis of task, terminal, transaction, time frame, and trace entry type. This complements the usefulness of auxiliary trace capturing large amounts of data.

Note: PRTCICSTRC formats part of the total available trace data. If the problem is to be passed to the IBM Support Center, both *USRSPC objects may be required. Use the CRTDUPOBJ CL command to copy these objects to a protected object for possible future use.

Prolog of a typical trace report

Here is an example of the prolog of a typical trace report. The prolog shows the options that were specified in the PRTCICSTRC command.

```
         V02R03M00                 Auxiliary Trace Table QTEST     /AEGADMRCA          Page      1
                                    * * *  P R O L O G  * * *
Auxiliary Trace Userspace:
  Object Name  . . . . . . . . . :   AEGADMRCA
  Library name . . . . . . . . . .   QTEST
Message/Data format  . . . . . . :   *FULL
Trace entry type . . . . . . . . :   *ALL
CICS/400 Transaction Id  . . . . :   SC01
CICS/400 Terminal Id . . . . . . :
CICS/400 Trace number  . . . . . :
CICS/400 Job number  . . . . . . :
Time period of trace:
  Start time and date:
      Start time . . . . . . . . :   *AVAIL
      Start date. . . . . . . . :    *BEGIN
  Ending time and date:
      End time . . . . . . . . . :   *AVAIL
      End date. . . . . . . . . :    *END
                                    Auxiliary Trace Table QTEST     /AEGADMRCA
```

Figure 7. Example of a trace table prolog

Information in the formatted CICS/400 trace entry

Each formatted trace entry (for an example, see Figure 8 on page 81) consists of the following pieces of information:

Count Provides a sequence number for each trace entry.

Date/time stamp
Shows when the trace entry was recorded.

Module Represents the CICS module recording the trace entry.

Type Indicates one of the six CICS trace entry types.

Function code Represents the internal CICS function number for the module.

Return code Indicates, for EXIT trace entries, the internal CICS return code. If the return code is all zeros, this indicates that the CICS module has successfully completed the function required and is returning normally to the previous module.

Trace number Represents the internal trace number for various events within a CICS module. This allows the tracking of the flow of control through a CICS module.

Resource Represents a CICS or OS/400 resource being used or about to be used by a CICS module. A typical use of this field is to refer to the key information for the various CICS resource tables.

This field may contain a file name, a program name, a transaction name, a terminal ID, an OS/400 program name, or other resource information.

Job number Represents the OS/400 job number in which the process is being executed.

Terminal ID Represents the four-character CICS terminal ID from which the transaction was initiated. This may be empty if the transaction is not a terminal-related transaction, or if the trace entry is being recorded by the control region.

Transaction ID
The CICS/400 transaction identifier.

Message data Represents special data recorded by the module. The data recorded pertains to the function being performed.

For USER trace entries, this is the data recorded by the application as the result of an EXEC CICS ENTER command.

Sample trace table

Following is a sample page of a typical trace table print.

Interpretation of the fields in a selected transaction from this table is shown in Figure 9 on page 83.

```
                                Auxiliary Trace Table QTEST     /AEGADMRCA          Page      3
COUNT   DATE   TIME        MODULE    TYPE   FUNCTION  RETCODE   TRNUM  RESOURCE   JOBNUM  TERM  TRAN  TASK
000931  102094 165228 133  AEGEIEIP  ENTRY  00000000  FFFFFFFF  0001  0001-00000  029412  TST1  ACCT  0294121
000932  102094 165228 134  AEGEIEIP  EVENT  00000000  FFFFFFFF  2001              029412  TST1  ACCT  0294121
Message/Data-
  0000 D9C5E3E4 D9D500E3 D9C1D5E2 C9C400                                        *RETURN.TRANSID.                *
000933  102094 165228 137  AEGEIEIP  EVENT  00000000  FFFFFFFF  2001              029412  TST1  ACCT  0294121
Message/Data-
  0000 C6E3D940 C2939683 92404040 40404040   40404040 40404040 40404040 40404040  *FTR Block                      *
  0020 C1C3F0F1 40404040 40404040 40404040   C1C3C3E3 E2C5E340 40404040 40404040  *AC01            ACCTSET        *
  0040 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                               *
       Lines  0060 to 0200 same as above.
  0220 0042FFFF 40404040 00000001 00000000   00000000 00000000 00014040 40404040  *....    ..................     *
  0240 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                               *
  0260 40404040 40404040 40404040 40404040   40404040 40404040 40400000 40404040  *                            ..   *
  0280 40404040 00004040 40404040 40404040   40404040 40404040 40400001 40404040  *    ..                      ..   *
  02A0 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040  *                               *
       Lines  02C0 to 0360 same as above.
000934  102094 165228 137  AEGEIEIP  EVENT  00000E08  FFFFFFFF  1000            00  029412  TST1  ACCT  0294121
000935  102094 165228 137  AEGPCRET  ENTRY  00000E08  FFFFFFFF  010B                029412  TST1  ACCT  0294121
000936  102094 165228 138  AEGPCRET  EXIT   00000E08  00000000  0244                029412  TST1  ACCT  0294121
000937  102094 165228 138  AEGEIEIP  EVENT  00000E08  00000000  1F00                029412  TST1  ACCT  0294121
Message/Data-
  0000 F0F0F0F1 60F0F0F0 F0F0F0F0 F060F0F0   F0F0F0F0 F0F060C6 C6C6C660 40404040  *0001-00000000-00000000-FFFF-   *
000938  102094 165228 139  AEGEIEIP  EXIT   00000E08  00000000  0002                029412  TST1  ACCT  0294121
Message/Data-
  0000 F0F0F0F1 60F0F0F0 F0F0F0F0 F060F0F0   F0F0F0F0 F0F060F0 F0F0F060 40404040  *0001-00000000-00000000-0000-   *
```

Figure 8. Sample trace table page

Each CICS module records, at a minimum, an ENTRY and EXIT trace pair for each invocation of the module. In addition, the module may record EVENT and ERROR trace entries to record significant events within the module.

Each CICS module is identified by the CICS/400 product prefix “AEG”, followed by a two-character CICS component code and a three-character suffix. Many of the component codes represent the CICS functional area in which the module resides; for example, “CS” for application shell, “CR” for the control region, or “TC” for terminal control. By looking at the module, it is relatively easy to discern which CICS functional area is recording the trace entry.

Interpreting the function code

The function code on the ENTRY or EXIT trace defines the function being requested by the module. If the first four characters of the formatted function code are zeros, then the last four characters represent the equivalent of an EXEC interface block function code (EIBFN). This indicates that an EXEC CICS command is being processed.

If the first four characters of the formatted function code are nonzero, and the last four characters contain an EIBFN code value, this indicates that CICS is internally requesting an EXEC Interface function to be performed.

If the first four characters are nonzero, and the last four characters do not represent an EIBFN value, the function code represents an internal CICS function code.

Interpreting the return code

If the formatted return code on an EXIT trace entry is zeros, the requested function has been processed successfully by the CICS module issuing the EXIT trace entry.

If the return code is not all zeros, the first two characters of the return code represent the internal CICS severity level: X'0A', X'14', X'1D', X'28' (decimal 10, 20, 30, 40) for the processing problem encountered. The next two characters represent the internal CICS numeric value for the component detecting the problem. The remaining four characters are the hexadecimal value representing the reason for the problem. If this number is in the range X'01' through X'63' (decimal 1 through 99), this represents the equivalent of an EIBRESP value as defined for CICS/400. If the value is greater than X'63' (decimal 99), then it is an internal CICS reason code.

See the *CICS for iSeries Application Programming Guide* for further details on EIBFN and EIBRESP values.

CICS modules may be able to recover correctly from severity level X'0A', X'14', and in some cases, level X'1D' return codes. However, ERROR trace entries are recorded for level X'1D' and X'28' return codes, which usually indicate an error that cannot be recovered from internally. This leads to **first failure data capture** routines being invoked. The job log for the task being executed, along with the CICS trace for that task provides problem determination information for the error recorded.

Typical transaction trace

For a typical transaction trace, assuming that auxiliary trace was active throughout the transaction's lifetime, a typical set of formatted trace entries would contain, but not be limited to:

- AEGCSRUN–Entry to start the transaction doing the PCT lookup and cause a link to the application program.
- AEGPCPGM–Entry to do the PPT lookup and call the application program. An EVENT trace with function X'00000E02' (LINK) is recorded just before calling the application program represented by the program name in the RESOURCE area.
- AEGEIEIP–The EXEC interface program writing an ENTRY and EXIT trace pair for the initial application program call to set up the EXEC interface environment.
- AEGEIEIP–A number of ENTRY and EXIT trace pairs for each execution of an EXEC CICS command from within the application program. Between the AEGEIEIP ENTRY and EXIT pair would be a series of AEGEIEIP EVENT trace entries followed by other CICS module trace entries for the functional area processing the request (for example, file control for an EXEC CICS READ command).

  The function code on the ENTRY trace record to the component modules indicates the type of EXEC interface function (EIBFN) being requested by the EXEC CICS command.
- A set of ENTRY and EXIT trace entries from AEGPCPGM for the EXEC CICS RETURN (function code X'00000E08') causing an end of the application program.
- AEGSPxxx trace entries recorded by the CICS syncpoint modules for task termination processing.
- AEGPCPGM–Recording an EVENT trace record on return from the application program.

- AEGPCPGM–Recording an EXIT trace entry on returning control to the application shell (function code X'00000E02').
- AEGCSRUN–Recording an EVENT entry on return from AEGPCPGM.

As indicated, the typical trace may contain many more trace entries than stated above, based on the type of the application, and whether the transaction processes normally or not.

Interpreting the trace table

Figure 9 highlights two trace entries from the auxiliary trace table previously shown in Figure 8 on page 81.

```
COUNT   DATE   TIME        MODULE    TYPE   FUNCTION  RETCODE   TRNUM  RESOURCE   JOBNUM  TERM  TRAN  TASK
000937  102094 165228 138  AEGEIEIP  EVENT  00000E08  00000000  1F00               029412  TST1  ACCT  0294121
Message/Data-
  0000 F0F0F0F1 60F0F0F0 F0F0F0F0 F060F0F0   F0F0F0F0 F0F060C6 C6C6C660 40404040  *0001-00000000-00000000-FFFF-    *
000938  102094 165228 139  AEGEIEIP  EXIT   00000E08  00000000  0002               029412  TST1  ACCT  0294121
Message/Data-
  0000 F0F0F0F1 60F0F0F0 F0F0F0F0 F060F0F0   F0F0F0F0 F0F060F0 F0F0F060 40404040  *0001-00000000-00000000-0000-    *
```

Figure 9. Sample trace table entries

Both of these trace entries contain information about the outcome of an EXEC CICS command which has been processed by the CICS API program AEGEIEIP. Let us look more closely at the AEGEIEIP EXIT trace record. The Count, Date, and Time fields show that this is entry number 938, and that it occurred on 20 October 1994, at 4:52 p.m., and 28.138 seconds.

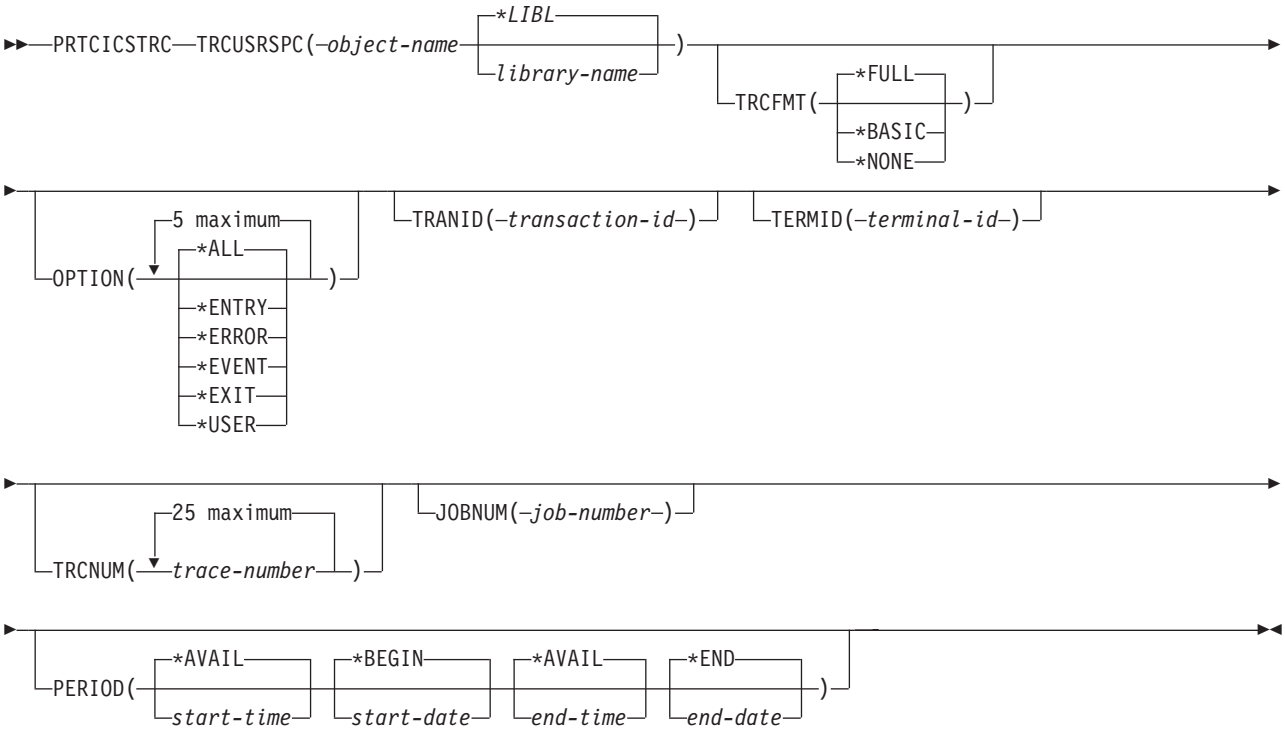
The first four positions of the function code are zeros. Following the explanation provided in "Interpreting the function code" on page 81, if the first four bytes of the function code are zero, the last four bytes represent an EIBFN. Consulting the EIBFN table in the *CICS for iSeries Application Programming Guide*, look up EIBFN 0E08. You will find the command being processed is EXEC CICS RETURN. The return code (RETCODE) in this example is 00000000 which indicates that the command has been successfully processed.

You can also see that this trace entry was recorded by transaction ACCT running on CICS terminal TST1 in OS/400 job 029412. Every trace entry contains this common level of information.

The AEGEIEIP EXIT trace entry also includes Trace Point Number (TRNUM), Resource and Message/Data information, which is unique to this particular trace entry. This additional information allows an experienced service representative to diagnose problems that may occur. The Message/Data in this example records the logical link level, EIBRESP, EIBRESP2, and Abend Code values returned to the application.

Chapter 13. PRTCICSTRC (Print CICS Trace) command

Job: B,I Pgm: B,I REXX: B,I Exec



PRTCICSTRC

PRTCICSTRC (Print CICS Trace) command

Purpose

The print CICS trace (PRTCICSTRC) command produces a job-oriented report from the CICS auxiliary trace user spaces. Entries are selected for inclusion in, or exclusion from, the report based on a variety of job details and interval times.

CL command defaults

The defaults given in the CL command description are those that are supplied with the iSeries system. You should check that your installation has not made any changes to these command default parameters.

Required parameters

TRCUSRSPC

The possible values for the CICS control region trace that is to be printed are:

Element 1

object-name: Specify the name of the auxiliary trace table user space object.

Element 2

Specify the name of the library where the CICS auxiliary trace user space object exists.

***LIBL:** The library list is used to locate the CICS auxiliary trace user object.

library-name: The name of the library where the auxiliary trace user space object exists.

Optional parameters

TRCFMT

Specifies the type of format that is produced for this report. Possible values are:

***FULL:** All trace data is printed up to a maximum of 4000 bytes.

***BASIC:** A maximum of 3 lines of 32 bytes each of trace data is printed.

***NONE:** No trace data is printed.

OPTION

Specifies the information being printed. Any combination, up to five, of the following may be selected.

***ALL:** All trace entries are selected for the report.

***ENTRY:** All trace entries marked as ENTRY are selected for the report.

***ERROR:** All trace entries marked as ERROR are selected for the report.

***EVENT:** All trace entries marked as EVENT are selected for the report.

***EXIT:** All trace entries marked as EXIT are selected for the report.

***USER:** All trace entries marked as USER are selected for the report.

TRANID

Specifies the transaction identifier that is to be used for producing the trace information.

TERMID
Specifies the terminal identifier that is to be used for producing the trace information.

TRCNUM
Specifies the trace numbers that are used for producing the trace information. Up to 25 may be specified.

JOBNUM
Specify the six-digit number of a job to select. All six digits must be specified (using leading zeros if necessary).

PERIOD
Specifies the period of time for which the auxiliary trace entries in the specified user space object are shown. This parameter contains two lists of two elements each.

```
PERIOD((start-time start-date)
       (end-time end-date))
```

If PERIOD is not specified, the following values are assumed.

```
PERIOD((*AVAIL *BEGIN) (*AVAIL *END))
```

Element 1: Starting time

One of the following values is used to specify the starting time at which or after which the auxiliary trace entries are collected. Data collected before the specified time and date is not included in the report.

***AVAIL:** The auxiliary trace entries in the specified user space object that are available for the specified starting date are shown.

start-time: Specify the starting time after which the data must be collected to be included in the report. The time is specified in 24-hour format and can be specified with or without a time separator:

- Without a time separator, specify a string of 4 or 6 digits (hhmm or hhmmss) where hh=hours, mm=minutes, and ss=seconds. Hours, minutes, and seconds must each be exactly 2 digits (using leading zeros if necessary).
- With a time separator, specify a string of 5 or 8 digits where the time separator specified for your job is used to separate the hours, minutes, and seconds. If you enter this command from the command line, the string must be enclosed in apostrophes. If a time separator other than the separator specified for your job is used, this command will fail.

Element 2: Starting date

One of the following values is used to specify the starting date on which or after which the auxiliary trace entries are collected. Data collected before this date is not included in the report.

***BEGIN:** Data records starting from the beginning of the trace file are included in the report.

start-date: Specify the starting date. The date must be specified in the job date format.

Element 3: Ending time

Use one of the following values to specify the ending time. Data collected after this time and date is not included in the report.

***AVAIL:** The logged data that is available for the specified ending date is shown.

end-time: Specify the ending time for the specified ending date that determines the auxiliary trace entries to be printed. See Element 1: Starting time for the formats in which time can be entered.

Element 4: Ending date

Use one of the following values to specify the date to end collection of the data records. Data collected after the specified date and time is not included in the report.

***END:** Data records through the last day and time of the collection period are included in the report.

end-date: Specify the ending date for which the auxiliary trace entries are to be printed. The date must be specified in the job date format.

Examples

```
PRTCICSTRC TRCUSRSPC(TRACEA) TRCFMT(*FULL)
```

This command will produce a full trace report for user space object TRACEA in the library list. A full report is produced with all of the available data in the object.

Part 4. Appendixes

Appendix A. Worksheet for transaction abends

This sample worksheet suggests what you should do when you have an abend code.

1. Record the abend code and messages

   Find the abend code from the user data of the transaction dump spool file and record any messages.

   _____ _____

2. Is this a CICS or USER abend code?

   For a CICS abend code, go to 3. If this is a USER abend code, check the application documentation and tell the appropriate person.

3. Look up the abend code

   If you need further advice go to 4.

4. Is this an ASRA abend?

   Yes; go to 6.
   No; go to 5.

5.

   Last Statement Executed _____

6. Analyze the data gathered.

   For most problems you should now have enough information to solve the problem. If you still cannot find the source, recheck the following:

   a. Parameters to or from other programs or systems
   b. Any needed resource that may not be available
   c. The formatted trace, for any unexplained flow
   d. The running environment, for any changes in it

Appendix B. Abend codes

When an abnormal condition occurs, the following message is sent to the job log where the transaction was being executed, and to the CSMT transient data destination (if defined):

transaction *tranid* **abnormal end** *abcode* **program** *progname*,

where:

- *tranid* is a CICS transaction identifier
- *abcode* is an abend code
- *progname* is a program name

See the *CICS for iSeries Administration and Operations Guide* for further information about defining the CSMT log.

Alternatively, the application can intercept abends by including an active EXEC CICS HANDLE ABEND command. The actual abend code can be discovered by issuing EXEC CICS ASSIGN ABCODE(ch).

All CICS transaction abend codes *abcode* are 4-character alphanumeric codes of the form:

`A`*`xxy`*

where:

- A is a IBM-assigned designation of a CICS transaction abend
- *xx* is a 2-character code assigned by CICS to further distinguish the error
- *y* is a 1-character alphanumeric code assigned by CICS

For each transaction abend code, the following information is provided:

- An explanation of events leading to or following the message
- The action that has been or will be taken by CICS (system action)
- The response recommended for the user or system operator
- The name of the module that determined that the message should be sent

ABM0

Explanation: The map specified for a BMS request could not be located in the specified map set.

System Action: The task is terminated with a transaction dump.

User Response: Ensure that:

- The correct map set object name is defined in the PPT
- The map set contains the required map definition
- The release level in the map is greater than the release level in the SIT

Module: AEGBMRMS, AEGBMSND

ABM3

Explanation: A BMS input or output request has been issued from a task that is not terminal-oriented.

System Action: The task is terminated with a transaction dump.

User Response: The task issuing the BMS request must be attached to a terminal.

Module: AEGBMSND

ABMB

Explanation: A cursor position was specified that is too large for the display device.

System Action: The task is terminated with a transaction dump.

User Response: Change the request to refer to a valid cursor position for the specified device.

Module: AEGBMSND

ABMI

Explanation: The map specified for a BMS input mapping request (an EXEC CICS SEND MAP command) was not an input map.

System Action: The task is terminated with a transaction dump.

User Response: Change the request to refer to an input map, or re-create the map to define it as an input or input/output map.

Module: AEGBMRMS

ABMO

Explanation: The map specified for a BMS output mapping request (an EXEC CICS SEND MAP command) was not an output map.

System Action: The task is terminated with a transaction dump.

User Response: Change the request to refer to an output map, or re-create the map to define it as an output or input/output map.

Module: AEGBMSND

ABMT

Explanation: The terminal type is not valid for a BMS request.

System Action: The task is terminated with a transaction dump.

User Response: Check the definition of the terminal in the TCT. Ensure that the device type is proper for the type of BMS request being processed.

Module: AEGBMRMS, AEGBMSND

ABMU

Explanation: The COBOL application program supplied a FROM area that has been corrupted. iSeries COBOL stores the pointer to each 01 level following the 01 level data area. If the data area is too small for incoming data, the pointer is destroyed.

System Action: The task is terminated with a transaction dump.

User Response: Investigate the 01 data area, and determine the correct size. If the size is correct, investigate the EXEC CICS command for an incorrect input or output size.

For example:

```
01 WS-DATA.
   03 WS-DATA1       PIC X(5).
   03 WS-DATA2       PIC X(10).

EXEC CICS WRITEQ TS QUEUE(TSQUE)
                    FROM(WS-DATA)
                    LENGTH(20)
END-EXEC.

MOVE 15 TO WS-LEN.

EXEC CICS READQ  TS QUEUE(TSQUE)
                    INTO(WS-DATA)
                    LENGTH(WS-LEN)
END-EXEC.

EXEC SEND MAP FROM(WS-DATA)
              MAP(WS-MAP)
END-EXEC.
```

The pointer to WS-DATA was overwritten when the EXEC CICS READQ TS command was executed, because the length of the output data was defined as 20 bytes, but the input area was defined as only 15 bytes.

Either reduce the length of the output queue to 15 bytes, or increase the size of the input area.

Module: AEGBMRMS, AEGBMSND

ABMX

Explanation: The area from which a map is to be received or sent has been corrupted. This abend may also occur if DBCS data is sent to a terminal which is defined to CICS/400 as being DBCS capable, but which cannot accept DBCS data.

System Action: The task is terminated with a transaction dump.

User Response: Review the program storage of the application program. Ensure that the 01 level immediately preceding the corrupted area is of the proper size.

Module: AEGBMSND

ADPL

Explanation: A request has been made from a Distributed Program Link program for a CICS service that is not allowed in a DPL program.

System Action: The task is terminated with a transaction dump.

User Response: Refer to the LINK command in the *CICS for iSeries Application Programming Guide* for the DPL–CICS command restrictions. Change the application program so that it does not use any CICS command in the restricted DPL subset.

Module: AEGBMRMS, AEGBMSND, AEGEXAID, AEGTCPAS

AEC1

Explanation: An attempt has been made to use the Temporary Storage Browse (CEBR), Command-Level Interpreter (CECI), Enhanced Master Terminal (CEMT), Execution Diagnostic Facility (CEDF), or the Resource Definition Online (CEDA) transaction on a terminal that is not supported for these transactions.

System Action: The task is terminated with a transaction dump.

User Response: Use a terminal that is supported by CEBR, CECI, CEMT, or CEDA. These transactions cannot be issued by APPC logical units or by IC start requests for the transaction, with the terminal defined as a printer. The terminal must be a display device.

Module: AEGBRTSS, AEGCIPGM, AEGDFFLG

AEXR

Explanation: In starting or ending an application program, CICS exception condition programs have been called internally with an invalid function code. CICS/400 First Failure Data Capture has logged the error.

System Action: The task is terminated with a transaction dump.

User Response: Perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problem (ANZPRB) commands to report the problem.

Module: AEGEXCLR, AEGEXINT, AEGEXTER

AEXZ

Explanation: A CICS command or supplied transaction has failed due to a catastrophic error. CICS/400 First Failure Data Capture has logged the error.

System Action: The task is terminated with a transaction dump. A CICS transaction dump is created.

User Response: Use the CICS transaction dump to try to determine the error. If auxiliary trace was active, use the PRTCICSTRC command to assist in problem determination. If no resolution can be found, perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problems (ANZPRB) commands. Contact the IBM Support Center.

Module: AEGUCFDC

AEIx, AEXx, AEYx

Explanation: An exception condition has occurred for which no EXEC CICS HANDLE CONDITION is active and the RESP or NOHANDLE option has not been included in the associated command. Because of their similar characteristics, the above-named abend codes are described as a group. Refer to the *CICS for iSeries Application Programming Guide* for the abend code and error conditions table.

System Action: The task is terminated with a transaction dump.

User Response: Change the application program in order to prevent the condition from recurring. Check the condition by using the RESP or NOHANDLE option, or by setting an EXEC CICS HANDLE CONDITION for the error. If necessary, the contents of the EIBRESP2 and the EIBRCODE fields in the EXEC Interface Block (EIB) may assist in determining the cause of the exception condition.

Module: AEGEIEIP

AEY9

Explanation: An invalid function code has been received by the EXEC Interface program.

System Action: The task is terminated with a transaction dump.

User Response: This occurs if:

- The CICS application program had been edited after translation and before compilation
- The program contains EXEC CICS INQUIRE or SET commands and the TGTRLS parameter of the CRTCICSCBL or CRTCICSC command is not set to *CURRENT

The code inserted by the translator has probably been changed incorrectly, causing the error.

Retranslate and recompile the program in error without editing the program after the translation step.

Module: AEGEIEIP

AEYQ

Explanation: A SYSIDERR condition was not handled.

System Action: The task is terminated with a transaction dump.

User Response: Check the application. The SYSIDERR condition is raised if a command uses a SYSID that is unknown or that specifies a link that is not currently available.

Module: AEGEIEIP

AEYZ

Explanation: The transaction routine request invokes APPC to connect to the remote program. APPC could not connect to the remote process.

System Action: The task is terminated with a transaction dump.

User Response: Examine the TCS entries to ensure that all connection entries are in service.

Module: AEGTRCAO, AEGTRFOB

AFCZ

Explanation: The transaction has issued a file control request for a file that has been defined to CICS as a file to be automatically journaled. In attempting to record the request to the CICS automatic journal, the journal write request has been unsuccessful.

System Action: The task is terminated with a transaction dump.

User Response: Examine the job log of the abending transaction to try to determine the problem with the CICS user journal file. Check the following problem list:

1. The user making the file control request is not authorized to the CICS user journal file.
   a. Authorize the user to the user journal file
   b. Change the FCT entry for the file to turn off automatic journaling for the file
2. The user journal file is full and is defined as nonswitchable. The journal write request could not be completed successfully. Before attempting the request again:
   a. Create a new user journal file allowing for more records in the file, or
   b. Change the FCT entry for the file to turn off automatic journaling for the file, or
   c. Change the JCT entry to a switchable journal

Module: AEGFCDRF, AEGFCRDN, AEGFCRDP, AEGFCREW, AEGFCWRI

AICB

Explanation: An interval control EXEC CICS RETRIEVE WAIT request has been issued during a CICS control region shutdown.

System Action: The task is terminated with a transaction dump.

User Response: None.

Module: AEGICRTV

AICH

Explanation: While performing an interval control request, the task was canceled before an internal GETMAIN request could be completed successfully.

System Action: The task is terminated with a transaction dump.

User Response: Examine the job logs of the abending task and its associated control region. Determine whether a short-on-storage condition has occurred. If so, either try the request again, or recycle the control region changing the SIT values for the CICS storage objects to allow for more storage. The AEG06xx series of messages in the job log indicate which space object's size should be expanded.

Module: AEGICDLA, AEGICPOS

AISB

Explanation: The mirror transaction (CSMI) has detected errors in the data passed to it from the attaching transaction.

System Action: The detecting mirror task is terminated with a transaction dump.

User Response: The invalid data is visible in the transaction dump. In addition, if CICS auxiliary trace was active at the time of the failure, a print of the trace using the PRTCICSTRC command will also show the invalid data.

Module: AEGFSMIR

AISD

Explanation: The mirror program executed a function shipping request and received a nonzero return code as a result. The data-flow control state of the intersystem link being used was such that the nonzero return code information could not be returned normally.

System Action: The detecting mirror task is terminated with a transaction dump.

User Response: The transaction dump provides information required to analyze the source of the nonzero return code at its point of origin. In addition, the job log of the mirror transaction job also contains information relevant to the error.

Module: AEGFSMIR

AISG

Explanation: The mirror program executed a function shipping request and produced a reply. The data-flow control state of the intersystem link being used was such that the reply could not be returned normally.

System Action: The mirror task is terminated with a transaction dump.

User Response: The transaction dump provides information required to analyze the source of the nonzero return code at its point of origin. In addition, the job log of the mirror transaction job also contains information relevant to the error.

Module: AEGFSMIR

AKC3

Explanation: A purge request was issued for the task.

System Action: The transaction is marked to be terminated with abend code AKC3.

User Response: None.

Module: AEGEIEIP

AKC6

Explanation: The maximum use count for an EXEC CICS ENQ command has been exceeded.

System Action: The task is terminated with a transaction dump.

User Response: The most likely cause is a program loop, where the program is doing repeated EXEC CICS ENQ commands for the same resource.

Examine the program causing the error. Ensure that the program is not looping as explained previously. Recompile the program, and try the transaction again.

Module: AEGKCENQ

AKCP

Explanation: While trying to allocate an outbound session to a remote CICS system, a read time-out error has occurred.

System Action: The task is terminated with a transaction dump.

User Response: The AKCP abend is a normal one. Coding the nonzero IDLETIME option in the transaction entry asks for the task to be abnormally terminated when this condition occurs.

The transaction should be rerun; the situation causing the suspend condition may have cleared itself up. The AKCP abend is expected occasionally, unless the transaction has a terminal read time-out of zero. No special action is necessary.

Module: AEGAPASO

AKCS

Explanation: A deadlock time-out condition has been detected. This condition can occur within a transaction that specifies a nonzero deadlock time-out in the WAITTIME option in the PCT, or when the NOSUSPEND option is used with EXEC CICS temporary storage queue or enqueue commands. The deadlock time-out occurs when a transaction has been waiting or suspended for longer than the time specified in the WAITTIME option.

System Action: The task is terminated with a transaction dump.

This abend can occur for a variety of internal CICS events, for example:

- Short-on-storage condition
- Inability to complete an EXEC CICS WRITEQ TS to MAIN storage due to a storage problem
- EXEC CICS ENQUEUE
- EXEC CICS ALLOCATE
- EXEC CICS RETRIEVE WAIT

User Response: The job log provides a message indicating what the suspended task was waiting on when the AKCS abend occurred. In addition, the job log of the control region may indicate the short-on-storage condition at the time of the request.

The transaction should be rerun; the situation causing the suspend condition may have cleared itself up. The AKCS abend is expected occasionally, unless the transaction has a deadlock time-out of zero. No special action is necessary.

Module: AEGAPASO, AEGAPRCV, AEGICRTV, AEGKCENQ

AKCT

Explanation: A terminal read time-out condition has been detected. This condition can occur when a transaction definition specifies a nonzero read time-out in the IDLETIME option in the PCT. The read time-out occurs when a transaction has been waiting for terminal input for a time longer than the time specified in the IDLETIME option.

System Action: The task is terminated with a transaction dump.

User Response: The AKCT abend is a normal one. Coding the nonzero IDLETIME option in the transaction entry asks for the task to be abnormally terminated when this condition occurs.

The transaction should be rerun; the situation causing the suspend condition may have cleared itself up. The AKCT abend is expected occasionally, unless the transaction has a terminal read time-out of zero. No special action is necessary.

Module: AEGTCPAR

APCR

Explanation: An invalid function code has been presented to program control.

System Action: The task is terminated with a transaction dump.

User Response: This is an internal CICS error. CICS First Failure Data Capture was called to log a problem in the OS/400 problem log. Perform problem analysis to report the problem to the IBM Support Center.

Module: AEGPCPGM

APCT

Explanation: One of the following has occurred:

1. An attempt to initiate a task for a transaction in the PCT failed when CICS could not link to the program specified as the transaction's initial program (PGMID) because:
   - The program entry in the PPT is DISABLED
   - The program entry in the PPT defines a map set
   - The program object referenced by the program entry in the PPT did not exist at, or has been deleted or changed since, Control Region startup
   - The requester is not authorized to the program object referenced by the program entry in the PPT
2. An attempt to load a map set failed because:
   - The map set entry in the PPT is DISABLED
   - The user space object referenced by the map set entry in the PPT did not exist at, or has been deleted since, Control Region startup
   - The requester is not authorized to the user space object referenced by the map set entry in the PPT

System Action: The task is terminated with a transaction dump.

User Response: If auxiliary trace was active at the time of the failure, format the trace for the transaction using the PRTCICSTRC command. The event and exit trace entries from AEGPCPGM provide information related to the error.

Examine the PPT entry using CEMT INQUIRE PROGRAM to determine the status of the program. Take the appropriate corrective action, based on the status of the PPT. Try the request again when the status problem is corrected.

Module: AEGPCPGM

APSR

Explanation: A CICS printer spooling module has been invoked with an invalid function code for printer spooling. The only valid function codes are the EIBFN codes for EXEC CICS SPOOLOPEN, SPOOLWRITE, and SPOOLCLOSE commands.

System Action: The task is terminated with a transaction dump.

User Response: This is an internal CICS error. CICS First Failure Data Capture was called to log a problem in the OS/400 problem log. Perform problem analysis to report the problem to the IBM Support Center.

Module: AEGPSCLS, AEGPSOPN, AEGPSWRT

APST

Explanation: A task issued an EXEC CICS SPOOLCLOSE, SPOOLOPEN, or SPOOLWRITE command, without either the NOHANDLE option or the RESP option.

System Action: The task is terminated with a transaction dump.

User Response: Correct the syntax of the command, specifying either the NOHANDLE option or the RESP option.

Module: AEGPSCLS, AEGPSOPN, AEGPSWRT

ARTA

Explanation: An attempt was made to perform a transaction routing request from a task that does not own a terminal device as the principal facility.

System Action: The task is terminated with a transaction dump.

User Response: Be sure that the transaction routing program AEGCRMTR has not been specified as the program for a transaction other than CRTE. Ensure that input to the CRTE transaction has not been initiated by a means other than terminal input, such as through the CICS/400 single-shot capability.

Module: AEGTRRTE

ARTB

Explanation: An attempt was made to perform a transaction routing request but there is no input TIOA, or the length of the input data is zero.

System Action: The task is terminated with a transaction dump.

User Response: Be sure that the transaction routing program AEGCRMTR has not been specified as the program for a transaction other than CRTE. Ensure that input to the CRTE transaction has not been initiated by a means other than terminal input, such as through the CICS/400 single-shot capability.

Module: AEGTRRTE

ASRA

Explanation: An abnormal error condition has occurred in a CICS application program, during the execution of a non-CICS statement.

System Action: The task is terminated with a transaction dump.

User Response: The transaction dump indicates which statement caused the exception. Correct the program, recompile and re-create the program object, and attempt the transaction again.

Module: AEGPCPGM

ASRB

Explanation: An abnormal error has occurred in a CICS application program during the execution of an EXEC CICS statement.

System Action: The task is terminated with a transaction dump.

User Response: The transaction dump indicates which EXEC CICS statement within the program caused the exception. Review all associated messages that were produced during this abnormal situation and perform the recovery indicated. Attempt the transaction again.

Module: AEGPCPGM

ASCP

Explanation: A storage control error has occurred.

System Action: The task is terminated with a transaction dump.

User Response: One of the following has occurred:

1. A GETMAIN request could not be completed. The task was purged either as a result of a CEMT transaction to purge the task, or after waiting longer that the deadlock time-out value specified for the transaction in the WAITTIME option of the PCT.

   If the task was purged via the CEMT transaction, this may have been an attempt to clear what appeared to be a deadlocked system.

   If the task timed out automatically, this may be due to insufficient main storage. Examine the job log of the control region and shell at the time of the abend to determine whether a short-on-storage condition was the cause of the problem. In addition, the WAITTIME option in the PCT could be increased to allow the task to wait longer.
2. A FREEMAIN request could not be completed. The application program has corrupted the storage accounting data for an area of storage previously obtained using an EXEC CICS GETMAIN command.

   Investigate the application program to determine why the program is writing data in storage beyond its allocated storage area.

Module: AEGSCFMA, AEGSCGMA

ASPX

Explanation: A CICS/400 transaction program has issued an EXEC CICS API call, but an activation-group-level commitment definition has been established for the activation group in which the transaction program is running. Programs associated with this activation group may not issue EXEC CICS API calls until this commitment definition is ended.

System Action: The task is terminated with a transaction dump.

User Response: Use the CL command WRKJOB OPTION(*CMTCTL) to display the current commitment-control status of the job. Make sure that CICS/400 transaction programs run in activation groups for which activation-group-level commitment definitions are not active.

Module: AEGSPRCI, AEGEIEIP

ATCV

Explanation: An application attempted an operation on a logical unit, but was not in the correct mode, for one of the following reasons:

1. CICS cannot perform the current request because another request is outstanding (EIBSYNC is set).
2. The application is communicating with an APPC system and is not in the correct state to perform the requested operation.

System Action: The task is terminated with a transaction dump.

User Response: If the first reason applies, change the application to issue a syncpoint and then issue the request.

If the second reason applies, issue the free request and reallocate the session.

When the cause of the problem has been ascertained, the application program should be changed to ensure that session-oriented information is acted upon before any further requests are sent across that session. The session status information is available in the EIBSYNC, EIBFREE, and EIBRECV fields immediately following the execution of send, receive, and converse requests. It is good programming practice to save them in application program storage after each request, for later testing.

Check the job log where the task was being executed for further explanatory messages issued by the APPC support functions within CICS.

Module: AEGAPFSM, AEGAPIAM, AEGAPICM, AEGAPIEM, AEGAPISM, AEGAPRCV, AEGAPSND, AEGAPWCM

ATND

Explanation: The APPC services of the OS/400 have determined that a CICS transaction should abnormally terminate. However, the CICS transaction is at a critical processing point. Immediate termination would put the CICS system at risk.

System Action: The task is terminated with a transaction dump, but only after it has reached a noncritical processing point.

User Response: Check the job log where the task was being executed for further explanatory messages issued by the APPC support functions within CICS.

Module: AEGTRXTP

ATNI

Explanation: A terminal error has occurred resulting in a TERMERR condition being set.

System Action: The task is terminated with a transaction dump.

User Response: There may be many causes for the terminal error, for example:

- An application program has built its own data stream which contains an invalid start buffer address or screen attribute character. The job log for the offending task should show a device response error at the time of the abend.
- A hardware failure may have occurred, preventing the completion of the terminal-related request. The job log of the offending task should show a device error for the offending task.

Examine the job log of the task for further explanatory messages issued by the system or by CICS related to the error. Correct the application in error and try the request again.

Module: AEGAPFSM, AEGAPWCM, AEGFSMIR, AEGFSMXP, AEGFSOUT, AEGTCPAR, AEGTCPAU, AEGTRFOB, AEGTRXTP

ATSP

Explanation: A task has issued an EXEC CICS WRITEQ TS command to a recoverable temporary storage queue and either:

1. The temporary storage queue name is currently being used by another task's unit of work, or
2. The task previously issued an EXEC CICS DELETEQ TS request and the task has not completed a logical unit of work.

System Action: The task is terminated with a transaction dump.

User Response: Depending on the cause, either:

1. Correct the application to avoid issuing multiple requests to the same recoverable queue, or
2. Correct the application to avoid issuing an EXEC CICS WRITEQ TS request to a recoverable queue, after an EXEC CICS DELETEQ TS request, before completing a unit of work.

Module: AEGTSWRQ

ATSS

Explanation: An input/output error has occurred while trying to complete a temporary storage request to auxiliary temporary storage.

System Action: The task is terminated with a transaction dump.

User Response: Examine the job log of the task where the abend occurred. The messages issued by the system and by CICS should provide an explanation for the problem and a possible resolution.

In addition, examine the job log of the control region. It is likely that the temporary storage file for auxiliary temporary storage have been either damaged, deleted, or no more records can be written to them.

Module: AEGTSWRQ

AXFB

Explanation: An unacceptable function management header (FMH) type has been found. The FMH must be type 05, 06, or 43.

System Action: The task is terminated with a transaction dump. CICS/400 First Failure Data Capture has logged the error.

User Response: Perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problem (ANZPRB) commands to report the problem.

Module: AEGFSMIR

AXFM

Explanation: The SYSIDERR condition has been raised during an attempt to daisy-chain a function shipping request. This can happen when the requested resource proves to be on yet another remote system.

System Action: The task is terminated with a transaction dump.

User Response: Ensure that the daisy-chaining of requests is intended and that all relevant intersystem links are in service.

Module: AEGFSXFP

AZCD

Explanation: An intersystem logic error has been detected during APPC processing. The length of the application data received (determined by the LL fields and the concatenation flags) does not match the length actually received. CICS has not received all the data that was expected.

System Action: The task is terminated with a transaction dump. CICS/400 First Failure Data Capture has logged the error.

User Response: Use the CICS transaction dump to try to determine the error. If auxiliary trace was active, use the PRTCICSTRC command to assist in problem determination. If no resolution can be found, perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problems (ANZPRB) commands. Contact the IBM Support Center.

Module: AEGTRXTP

AZTC

Explanation: An error response was received from the CICS transformer program when attempting to perform transformation 2.

System Action: The task is terminated with a transaction dump. CICS/400 First Failure Data Capture has logged the error.

User Response: Use the CICS transaction dump to try to determine the error. If auxiliary trace was active, use the PRTCICSTRC command to assist in problem determination. If no resolution can be found, perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problems (ANZPRB) commands. Contact the IBM Support Center.

Module: AEGTRAOR, AEGTRCAO, AEGTRSCH

AZTE

Explanation: An error response was received from the CICS transformer program when attempting to perform transformation 4.

System Action: The task is terminated with a transaction dump. CICS/400 First Failure Data Capture has logged the error.

User Response: Use the CICS transaction dump to try to determine the error. If auxiliary trace was active, use the PRTCICSTRC command to assist in problem determination. If no resolution can be found, perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problems (ANZPRB) commands. Contact the IBM Support Center.

Module: AEGTRAOR, AEGTRDWE

AZTL

Explanation: An attempt was made to run a CICS task on a remotely owned terminal that cannot be used to run the transaction.

System Action: The task is terminated with a transaction dump.

User Response: Check the terminal control table entries on all the CICS systems involved to ensure that the terminal requested can execute the desired transaction.

Module: AEGTRFIB, AEGTRFOB

AZTV

Explanation: An invalid FMH (no FMH 43) has been received from the remote system.

System Action: The task is terminated with a transaction dump. CICS/400 First Failure Data Capture has logged the error.

User Response: Use the CICS transaction dump to try to determine the error. If auxiliary trace was active, use the PRTCICSTRC command to assist in problem determination. If no resolution can be found, perform problem analysis using the Work with Problem (WRKPRB) or Analyze Problems (ANZPRB) commands. Contact the IBM Support Center.

Module: AEGTRAOR, AEGTRCAO, AEGTRFIB, AEGTRFOB

AZTW

Explanation: An attempt was made:

- To attach a task on a remotely owned terminal that was already running a task
- To install a terminal on a remotely-owned region when an earlier delete request for that terminal has not yet terminated

System Action: The task is terminated with a transaction dump.

User Response: Check the terminal control table entries on all the CICS systems involved to ensure that the terminal requested can execute the desired transaction.

Module: AEGTRFIB

Appendix C. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator
3605 Highway 52 N
Rochester, MN 55901-7829
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

CICS
CICS/400
COBOL/400
e (Stylized)
IBM

iSeries
iSeries 400
Operating System/400
OS/400
Redbooks
RETAIN
SAA
SystemView
400

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Index

R

S

T

U

W

Readers' Comments — We'd Like to Hear from You

iSeries
CICS for iSeries Problem Determination
Version 5

Publication No. SC41-5453-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC41-5453-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
ATTN DEPT 542 IDCLERK
3605 HWY 52 N
ROCHESTER MN 55901-7829

Fold and Tape | **Please do not staple** | Fold and Tape

SC41-5453-00

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC41-5453-00